

# NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



DTIC QUALITY INSPECTED 4

## THESIS

### OPTIMAL DESIGN OF NONLINEAR SHOCK ISOLATION FOR LARGE, LOCALLY NONLINEAR STRUCTURAL SYSTEMS

by

Brian R. Durant

December 1998

Thesis Advisor:

Joshua H. Gordis

Approved for public release; distribution is unlimited.

19990219027

**REPORT DOCUMENTATION PAGE**

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

<b>1. AGENCY USE ONLY (Leave blank)</b>		<b>2. REPORT DATE</b> December 1998	<b>3. REPORT TYPE AND DATES COVERED</b> Master's Thesis
<b>4. TITLE AND SUBTITLE:</b> Optimal Design of Nonlinear Shock Isolation for Large, Locally Nonlinear Structural Systems		<b>5. FUNDING NUMBERS</b>	
<b>6. AUTHOR(S)</b> Durant, Brian R.			
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Naval Postgraduate School Monterey CA 93943-5000		<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b>		<b>10. SPONSORING/MONITORING AGENCY REPORT NUMBER</b>	
<b>11. SUPPLEMENTARY NOTES</b> The views expressed here are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
<b>12a. DISTRIBUTION/AVAILABILITY STATEMENT</b> Approved for public release; distribution is unlimited.		<b>12b. DISTRIBUTION CODE</b>	
<b>13. ABSTRACT (maximum 200 words)</b> The method of Time Domain Structural Synthesis is reviewed and examples of linear structural modification and sub-structure coupling are presented. The general formulation for both linear and nonlinear syntheses are compared to highlight the similarities between the two and the difficulties experienced with the introduction of nonlinear elements. Examples of linear and nonlinear sub-structural coupling are presented to emphasize the similarity of the procedures used. Finally, a general approach for the optimal design of nonlinear shock isolation for large structural systems is developed. The repeated solution of arbitrarily large finite element models with localized nonlinear components is made possible through the use of a highly efficient nonlinear transient re-analysis procedure. The method has demonstrated order of magnitude decreases in compute time over traditional direct integration methods and is exact, regardless of the nature of the isolation. An example is included which demonstrates the procedure.			
<b>14. SUBJECT TERMS</b> Optimization, Nonlinear Isolation, Shock Isolation, Nonlinear Shock Isolation, Optimization of Nonlinear Shock Isolation		<b>15. NUMBER OF PAGES</b> 103	
		<b>16. PRICE CODE</b>	
<b>17. SECURITY CLASSIFICATION OF REPORT</b> Unclassified	<b>18. SECURITY CLASSIFICATION OF THIS PAGE</b> Unclassified	<b>19. SECURITY CLASSIFICATION OF ABSTRACT</b> Unclassified	<b>20. LIMITATION OF ABSTRACT</b> UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)  
Prescribed by ANSI Std. Z39-18 298-102



Approved for public release; distribution is unlimited.

**OPTIMAL DESIGN OF NONLINEAR SHOCK ISOLATION  
FOR LARGE, LOCALLY NONLINEAR STRUCTURAL SYSTEMS**

Brian R. Durant  
Lieutenant, United States Navy  
B.S. Physics, Rensselaer Polytechnic Institute, 1992

Submitted in partial fulfillment of the  
Requirements for the degree of

**MASTER OF SCIENCE IN MECHANICAL ENGINEERING**

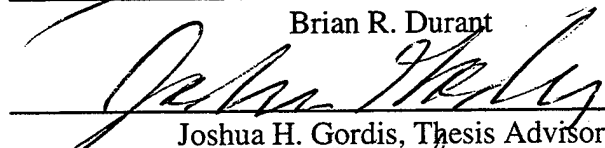
from the

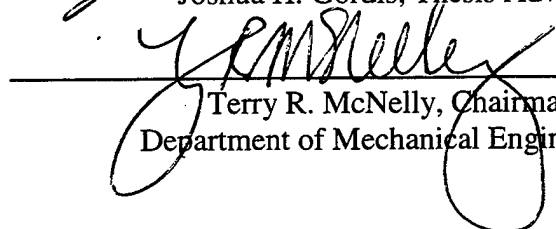
**NAVAL POSTGRADUATE SCHOOL  
December 1998**

Author:

  
Brian R. Durant

Approved by:

  
Joshua H. Gordis, Thesis Advisor

  
Terry R. McNelly, Chairman  
Department of Mechanical Engineering



## **ABSTRACT**

The method of Time Domain Structural Synthesis is reviewed and examples of linear structural modification and sub-structure coupling are presented. The general formulation for both linear and nonlinear syntheses are compared to highlight the similarities between the two and the difficulties experienced with the introduction of nonlinear elements. Examples of linear and nonlinear sub-structural coupling are presented to emphasize the similarity of the procedures used. Finally, a general approach for the optimal design of nonlinear shock isolation for large structural systems is developed. The repeated solution of arbitrarily large finite element models with localized nonlinear components is made possible through the use of a highly efficient nonlinear transient re-analysis procedure. The method has demonstrated order of magnitude decreases in compute time over traditional direct integration methods and is exact, regardless of the nature of the isolation. An example is included which demonstrates the procedure.



## TABLE OF CONTENTS

I. INTRODUCTION .....	1
II. INTEGRAL EQUATION FORMULATION FOR TRANSIENT STRUCTURAL SYNTHESIS .....	3
A. STRUCTURAL MODIFICATION.....	3
B. SUBSTRUCTURE COUPLING.....	6
C. SOLUTION OF THE VOLTERRA INTEGRAL EQUATION .....	8
D. EXAMPLES .....	11
III. NONLINEAR SYNTHESIS .....	17
A. GENERAL (LINEAR) SYNTHESIS FORMULATION .....	17
B. NONLINEAR SYNTHESIS FORMULATION .....	21
C. EXAMPLES .....	23
IV. OPTIMIZATION .....	27
A. OPTIMIZATION ROUTINE FORMULATION.....	28
1. Calculation of Objective Function .....	29
2. Calculation of Constraints .....	30
3. Design Variables .....	31
B. OPTIMIZATION COMPUTER CODE.....	32
C. EXAMPLE .....	35
V. CONCLUSIONS AND RECOMMENDATIONS.....	49
A. CONCLUSIONS .....	49
B. RECOMMENDATIONS.....	51
APPENDIX A. TIME-DOMAIN SYNTHESIS COMPUTER CODES FOR EXAMPLES 2-1 AND 2-2.....	53
APPENDIX B. LINEAR AND NONLINEAR SYNTHESIS COMPUTER CODE FOR EXAMPLES 3-1 AND 3-2.....	61
APPENDIX C. OPTIMIZATION COMPUTER CODES .....	77
LIST OF REFERENCES .....	85
INITIAL DISTRIBUTION LIST .....	87





## LIST OF FIGURES

Figure 2-1. General NDOF System.....	4
Figure 2-2. General Substructure Coupling .....	6
Figure 2-3. One Degree Of Freedom Spring-Mass System. ....	11
Figure 2-4. Transient Response of a One DOF System with a Stiffness Modification .....	13
Figure 2-5. Multi-DOF System with a Stiffness Modification at DOF #2.....	14
Figure 2-6. Synthesized Transient Response of DOF #4 to a Stiffness Modification at DOF #2.....	15
Figure 3-1. General NDOF System.....	18
Figure 3-2. Substructure Coupling/Modification for Nonlinear Synthesis .....	22
Figure 3-3. Mass-Plate System Experiencing Base Excitation .....	23
Figure 3-4. Nonlinear Spring Force-Displacement Characteristic .....	24
Figure 3-5. Base Excitation $Y(t)$ (Blast Function) .....	25
Figure 3-6. Transient Response of Computer with Linear Spring and Damper Modification	25
Figure 3-7. Transient Response of Computer with Nonlinear Spring and Linear Damper Modification.....	26
Figure 4-1. Sample Nonlinear Force-Displacement Curve .....	33
Figure 4-2. Optimization Routine Flowpath .....	34
Figure 4-3. Plate-Mass System Experiencing Base Excitation .....	35
Figure 4-4. Starting Point Force-Displacement Characteristic .....	37
Figure 4-5. Comparison of Acceleration VS Time Between Starting Point and Optimal Design Modifications.....	40
Figure 4-6. Comparison of Displacement VS Time Between Starting Point and Optimal Design Modifications.....	41
Figure 4-7. Change in Stiffener Slope (Base Isolator) VS Iteration Number .....	42
Figure 4-8. Base Damper Characteristic Slope VS Iteration Number .....	43
Figure 4-9. Objective Function Value VS Iteration Number .....	44
Figure 4-10. Optimal Design Acceleration VS Time Response .....	45
Figure 4-11. Optimal Design Displacement VS Time Response .....	45



## LIST OF TABLES

Table 4-1. Optimization Routine MATLAB Generated Output .....	37
Table 4-2. Optimization Start Point Inputs .....	39
Table 4-3. Comparison of Starting Point Values VS Time Required .....	47



## ACKNOWLEDGMENTS

I would like to express my sincerest appreciation to Professors Joshua H. Gordis and Young S. Shin for their dedicated support throughout this endeavor. Their wisdom and technical guidance significantly enhanced my education at the Naval Postgraduate School.

I would also like to thank my wife, Katherine, for her patience, support, and understanding. She has made my effort infinitely more enjoyable. And finally, to my daughter, Colleen, who has taught me that play is often times the best way to solve any problem.



## I. INTRODUCTION

With the introduction of Commercial-Off-the-Shelf (COTS) technology onto military platforms, comes the need to ensure that such equipment is properly isolated from shock. Since COTS are not designed to meet military specifications with regard to shock survivability, shock isolation systems must be developed which ensure equipment survivability in a combat environment. However, to physically test each equipment configuration until a suitable mount is found would become prohibitively expensive. A more cost-effective approach involves modeling equipment and mounting systems on a computer. Computer simulations could then be conducted to determine the feasibility of different mounting configurations.

Using finite element (FE) techniques and powerful computers, it is now possible to construct mathematical models and conduct response analysis of complex structures. By "breaking down" a complex system into its most basic components (stiffness, masses, and dampers), the behavior of the system may be determined through the solution of a series of less complex equations. Assembling the "pieces" of the model into a set of system matrices allows for examination of structural responses (static and dynamic) of the complete system. The final phase of this process, solution of the model responses, is the most computationally demanding step in the FE process.

Having completed a response analysis, engineers may wish to make modifications (small or large) to the system, thus requiring further simulation. Traditionally, this would require the engineer to repeat the entire process for each change desired. However, an optimization routine that would determine the optimal design for a given structure as well as perform the Finite Element Analysis (FEA) for each "design" would be far more preferable. Even this method is not the most desirable solution, though, since the entire model must be reconstructed for each change.

A more efficient method, that requires a system model to be constructed only once, and allows for solution of relatively few system elements, has been developed. The method dealt with in this thesis involves the use of synthesis procedures to generate post-



modification system responses. This synthesis method (Time-Domain Structural Synthesis (References [1] and [2])) uses pre-modification model data, structural modifications made (in the form of stiffness, mass or damping changes), and a series of equations to solve for the modified system response. In using this method, the engineer need only solve for the large, pre-modification model a single time. Once this solution is obtained, only the modification matrices need be developed. Specifically, time-domain synthesis requires the pre-modification system's impulse response functions (IRF's), and the convolution integral to solve for post-modification response. The governing equation of time-domain synthesis is a non-standard, non-homogeneous Volterra Integrodifferential Equation (VIDE) of the second kind. This VIDE is solved numerically in order to calculate the post-modification transient response. Thus, this technique allows for considerable reduction in processor time and improved capability for the application of optimization routines.

Thus, while the FEA method requires a reconstruction of the entire model for each modification made, the Time-Domain Structural Synthesis method allows for a single, pre-modification model to be built. The response characteristics of this pre-modification model are determined one time. This response characteristic is then used repeatedly to solve for the transient response of as many mounting configurations as desired. The timesavings achieved over FEA method in this way alone may be considerable. Combine this with the fact that the FEA method requires solution of the transient response of the entire model while Time-Domain Synthesis requires solution of an arbitrarily small number of elements, and the foundation of optimization of large structures is built.

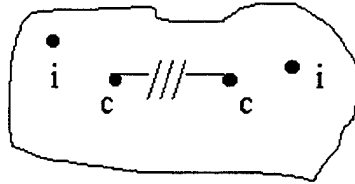
## II. INTEGRAL EQUATION FORMULATION FOR TRANSIENT STRUCTURAL SYNTHESIS

The method of structural synthesis allows the user to calculate the dynamic response of two or more substructures coupled together. It also allows for solutions to systems in which individual structural members may have been added, removed, or modified in some way. As stated above, the motivation to use this method lies in the computational savings of solving only for the synthesized model rather than the, often much larger, completely assembled model. A time domain, structural synthesis allows for the reduction of a model such that only the coordinates of interest are retained. It also allows for an exact synthesis of system damping. Perhaps most importantly, time domain synthesis allows for solution of the modified system using pre-modification transient response data. Thus, solutions of the modified system equations need not be obtained. Regardless of the modifications made to the original system, the same baseline, pre-modification transient responses are used to generate the synthesized response. This allows for more rapid solution of the multiple iterations often required in an optimization routine. The material presented in this section is simply an overview of the background work of Reference [1] and serves as a foundation for objective of this thesis. This material is presented to give the reader a general understanding of the techniques used in synthesis routines, both linear and nonlinear, which follow.

### A. STRUCTURAL MODIFICATION

Consider an arbitrary structure, as in Figure 2-1, which may be partitioned into two subsets of physical coordinates. For simplicity, examples programs and calculations were carried out using one and four degree-of-freedom (translation only), spring-mass models. Modifications were made only to spring stiffness (linear modifications only), with no changes in mass or addition of dampers. In each of these models those coordinates where modifications are installed are designated as Connection Coordinates and are represented by  $\{x_c\}$ . Coordinates that are not directly involved in the structural

modification, but whose post-modification transient response are of interest to the user, are designated as Internal Coordinates, and are represented by  $\{x_i\}$ .



**Figure 2-1: General NDOF System**

The following is an overview of the derivation of the equations used to perform post-modification response analysis (linear and nonlinear) after structural modification. A more detailed derivation is available in Ref. [1], and shall not be included here.

Starting with the convolution integral, the total dynamic response of the system can be expressed as follows:

$$\begin{Bmatrix} x_i(t) \\ x_c(t) \end{Bmatrix} = \begin{Bmatrix} x_i(t) \\ x_c(t) \end{Bmatrix}_h + \int_0^t \begin{bmatrix} h_{ii}(t-\tau) & h_{ic}(t-\tau) \\ h_{ci}(t-\tau) & h_{cc}(t-\tau) \end{bmatrix} \begin{Bmatrix} F_i(t) \\ F_c(t) \end{Bmatrix} d\tau \quad (2.1)$$

Where the  $[h(t)]$  are matrices of impulse response functions (IRF's), while the subscript h denotes the homogenous solution. The force vector  $\{F\}$  contains forces experienced by both the internal and connection DOF. Internal DOF will experience externally applied forces (denoted by superscript e) only. Connection DOF, however, will experience externally applied forces as well as reaction forces caused by the

modifications (denoted by superscript \*) associated with the respective DOF. Or, in vector notation:

$$\begin{Bmatrix} F_i(\tau) \\ F_c(\tau) \end{Bmatrix} = \begin{Bmatrix} F_i^e(\tau) \\ F_c^e(\tau) \end{Bmatrix} + \begin{Bmatrix} 0 \\ F_c^*(\tau) \end{Bmatrix} \quad (2.2)$$

Thus, we can express the synthesized response as

$$\begin{Bmatrix} x_i(t) \\ x_c(t) \end{Bmatrix}^* = \begin{Bmatrix} x_i(t) \\ x_c(t) \end{Bmatrix} + \int_0^t \begin{bmatrix} h_{ic}(t-\tau) \\ h_{cc}(t-\tau) \end{bmatrix} \begin{Bmatrix} F_i^e(\tau) \\ F_c^e(\tau) \end{Bmatrix} d\tau \quad (2.3)$$

In the case of Equation (2.3), the superscript \* denotes the synthesized quantity (displacement in this case). Also, note that the first term on the right hand side of Equation (2.3) is no longer simply the homogeneous response, but rather

$$\begin{Bmatrix} x_i(t) \\ x_c(t) \end{Bmatrix} = \begin{Bmatrix} x_i(t) \\ x_c(t) \end{Bmatrix}_h + \int_0^t \begin{bmatrix} h_{ii}(t-\tau) & h_{ic}(t-\tau) \\ h_{ci}(t-\tau) & h_{cc}(t-\tau) \end{bmatrix} \begin{Bmatrix} F_i^e(t) \\ F_c^e(t) \end{Bmatrix} d\tau \quad (2.4)$$

which is a nonstandard, non-homogeneous Volterra Integrodifferential Equation (VIDE) of the second kind.

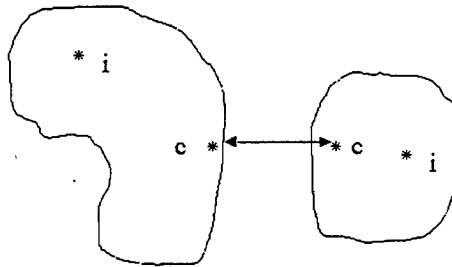
Once the synthesized, transient solution of the connection DOF has been determined, the transient response of the internal DOF may be calculated using the remaining portions of Equation (2.3).

## B. SUBSTRUCTURE COUPLING

Still using Equation (2.1), where the subscripts c and i retain their original meaning, a governing equation for the solution of synthesized transient response under the conditions of substructure coupling shall be developed.

Given two substructures, a and b (as in Figure (2.2)) the coordinates for each can be partitioned as

$$\{x_c(t)\} = \begin{bmatrix} x_c^a & x_c^b \end{bmatrix}^T \quad \{x_i(t)\} = \begin{bmatrix} x_i^a & x_i^b \end{bmatrix}^T \quad (2.5)$$



**Figure 2-2: General Substructure Coupling**

Just as with structural modification, internal DOF experience external forces only, while connection DOF experience the additional, reaction forces due to coupling. The force vector now becomes

$$\{F_c^*(t)\} = [R]\{\tilde{F}_c^*(t)\} \quad (2.6)$$

where  $[R]$  (which reflects force equilibrium) is the Boolean matrix (each column of  $[R]$  is all zero except for a 1 and a -1 in rows corresponding to coupled coordinates). The synthesized response now becomes, after pre-multiplying by  $[R]^T$ ,

$$\{\tilde{x}_c^*(t)\} = \{\tilde{x}_c(t)\} + \int_0^t [\tilde{h}_{cc}(t-\tau)]\{\tilde{F}_c^*(\tau)\}d\tau \quad (2.7a)$$

where

$$\{\tilde{x}_c(t)\} = [R]^T\{x_c(t)\} \quad (2.7b)$$

and

$$[\tilde{h}_{cc}(t)] = [R]^T[h_{cc}(t)][R] \quad (2.7c)$$

Finally, what is left, is a homogeneous, linear Volterra Integral Equation of the first kind for the coupling force vector. In order to generate a solution for  $\{\tilde{F}_c^*(t)\}$  two derivatives are taken, resulting in

$$[\dot{\tilde{h}}_{cc}(0)]\{\tilde{F}_c^*(t)\} = -\{\ddot{\tilde{x}}_c(t)\} - \int_0^t [\ddot{\tilde{h}}_{cc}(t-\tau)]\{\tilde{F}_c^*(\tau)\}d\tau \quad (2.8)$$

which may be solved numerically for the coupling force vector. This is substituted into Equation (2-3), that provides a solution for the synthesized, transient response of the coupled system.

### C. SOLUTION OF THE VOLTERRA INTEGRAL EQUATION

A more detailed derivation of the solutions to Volterra integral equations is presented in References [1] and [3], a general summary is presented here.

$$h(x)u(x) = f(x) + \int_a^{b(x)} K(x, \xi)u(\xi)d\xi \quad (2.9)$$

Equation (2.9) represents the most general, linear integral equation in  $u(x)$ . This becomes the Volterra Integral Equation when  $b(x)=x$ . Should  $h(x)=0$  (not to be confused with the  $h(t)$  for IRF's), this equation becomes the Volterra Integral Equation of the first kind. However, if  $h(x)=1$ , the result is a Volterra Integral Equation of the second kind.

Since, as shown in Reference [3], any general initial value problem may be converted into a Volterra Integral Equation of the second kind, the solution of the response of a spring-mass-damper system readily lends itself to the Volterra Equations:

$$([M])\frac{d^2x(t)}{dt^2} + ([C])\frac{dx(t)}{dt} + ([K])x(t) = F(t) \quad (2.10)$$

In order to solve a VIDE of the second kind, the integration interval is first divided into a set of  $n$ -subintervals with time step  $\Delta t = (x_n - a)/n$ , where  $x_n$  is the end point chosen for  $x$ . Setting  $t_0 = a$  and  $t_j = a + j\Delta t = t_0 + \Delta t$ , the value of the solution at  $t_i$  or  $x_i$  will be referred to as  $u(t_i) \equiv u(x_i) \equiv u_i$ ,  $f(x_i) = f_i$ . Similarly, the value of the Kernel  $K(x, t)$  at  $(x_i, t_i)$  will be  $K(x_i, t_i) \equiv K_{ij}$ . Using the trapezoidal rule with  $n$ -subintervals, the integral is approximated as

$$\int_a^x K(x,t)u(t)dt \approx \Delta t \left[ \frac{1}{2} K(x,t_0)u(t_0) + K(x,t_1)u(t_1) + \dots + K(x,t_{n-1})u(t_{n-1}) + \frac{1}{2} K(x,t_n)u(t_n) \right] \quad (2.11)$$

$$\Delta t = \frac{t_j - a}{j} = \frac{x - a}{n}, \quad t_j \leq x, \quad j \geq 1, \quad x = x_n = t_n$$

which may be approximated by the summation

$$u(x) = f(x) + \Delta t \left[ \frac{1}{2} K(x,t_0)u(t_0) + K(x,t_1)u(t_1) + \dots + \dots + K(x,t_{n-1})u(t_{n-1}) + K(x,t_n)u(t_n) \right] \quad (2.12)$$

$$t_j \leq x, \quad j \geq 1, \quad x = x_n = t_n$$

Substituting for each time interval, simplifying, and transferring solution ( $u_i$ ) terms to the left and non-homogeneous ( $f_i$ ) terms on the right, the following triangular system of equations results.

$$\begin{array}{cccccccc} u_0 & \dots & \dots & \dots & \dots & \dots & \dots & = f_0 \\ -\frac{\Delta t}{2} K_{10} u_0 + \left(1 - \frac{\Delta t}{2} K_{11}\right) u_1 & \dots & \dots & \dots & \dots & \dots & \dots & = f_1 \\ -\frac{\Delta t}{2} K_{20} u_0 - \Delta t K_{21} u_1 + \left(1 - \frac{\Delta t}{2} K_{22}\right) u_2 & \dots & \dots & \dots & \dots & \dots & \dots & = f_2 \\ -\frac{\Delta t}{2} K_{30} u_0 - \Delta t K_{31} u_1 - \Delta t K_{32} u_2 + \left(1 - \frac{\Delta t}{2} K_{33}\right) u_3 & \dots & \dots & \dots & \dots & \dots & \dots & = f_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{\Delta t}{2} K_{n0} u_0 - \Delta t K_{n1} u_1 - \Delta t K_{n2} u_2 - \dots + \left(1 - \frac{\Delta t}{2} K_{nn}\right) u_n & \dots & \dots & \dots & \dots & \dots & \dots & = f_n \end{array} \quad (2.13)$$



Note that the  $K_{ij}$  elements equal zero in Equation (2.13) due to  $h_{ij}(0)=0$ . Also note that the solutions are obtained by forward substitution, starting with  $u_0=f_0$  in the first equation being substituted into the second equation to find  $u_1$ . The results of the second equation are used to find the solution of the third and so on until  $u_n$  is solved.

In the case of structural modification the governing equation is

$$\{x_c^*(t)\} = \{x_c(t)\} - \int_0^t [h_{cc}(t-\tau)] \left\{ -[M^*]\{\ddot{x}^*(\tau)\} - [C^*]\{\dot{x}^*(\tau)\} - [K^*]\{x^*(\tau)\} \right\} d\tau \quad (2.14)$$

where, for a simple stiffness modification, the equation reduces to

$$\{x_c^*(t)\} = \{x_c(t)\} + \int_0^t [h_{cc}(t-\tau)] \{[K^*]\{x^*(\tau)\}\} d\tau. \quad (2.15)$$

The  $\{x_c(t)\}$  is the pre-modification system response, the IRF matrix  $[h_{ic}(t)]$  represents the Kernel matrix, and the  $\{x_c^*(t)\}$  vector is the desired, post-modification system response. As with the derivation above, the solution is obtained through forward substitution as follows

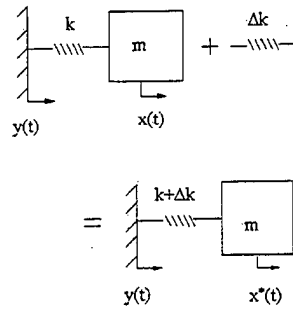
$$\{x_c^*(t)\} = [KERNEL]^{-1} \{x_c(t)\}. \quad (2.16)$$

As mentioned previously, structural modification already presents itself in the form of a VIDE of the second kind. Structural coupling, however, is in the form of a VIDE of the first kind. In order to use the methods described above, the governing equations for structural coupling must be reduced to a VIDE of the second kind. This can be accomplished by following the procedure outlined in Reference [3].

Thus, the Volterra Integral Equations of the second kind allow for a less complicated solution of unwieldy and complex matrix equations. The VIDE of the second kind becomes the basis of the Time-Domain synthesis solution.

## D. EXAMPLES

### Example 1. One Degree of Freedom Stiffness Modification



**Figure 2.3** One Degree Of Freedom Spring-Mass System

Starting with the simplest possible modification, consider the single DOF spring-mass system shown in Figure 2.3. A spring-mass system is modified by increasing the stiffness (linear modification) of the spring to ground by some value,  $\Delta k$ . The synthesized transient response is generated by using the transient response of the original system in conjunction with its impulse response function (IRF). Given a harmonic forcing function of the form  $F \sin \Omega t$  (where  $F$  is the amplitude and  $\Omega$  is the forcing frequency), the solution to this system's equation of motion ( $m\ddot{x} + kx = F \sin \Omega t$ ) is readily calculated as

$$x(t) = \frac{F}{\omega_n^2 - \Omega^2} \left( \sin \Omega t - \frac{\Omega}{\omega_n} \sin \omega_n t \right) \quad (2.17)$$

and the general form for the IRF is

$$h(t) = \sum_{p=1}^r \phi_i^p \phi_j^p t + \sum_{p=r+1}^n \frac{\phi_i^p \phi_j^p}{\omega_{dp}} e^{-\zeta_p \omega_{np} t} \sin(\omega_{dp} t) \quad (2.18)$$

where:  $r$  - number of rigid body modes

$n$  - number of modes retained

$\phi_i^p, \phi_j^p$  - the  $p^{\text{th}}$  modal response of the  $i^{\text{th}}, j^{\text{th}}$  mode, respectively

$\omega_{dp}$  - the  $p^{\text{th}}$  damped natural frequency

$\zeta_p$  - the  $p^{\text{th}}$  damping factor

$\omega_{np}$  - the  $p^{\text{th}}$  natural frequency

$t$  - time in seconds

which reduces to

$$h(t) = \frac{\sin(\omega_n t)}{\omega_n} \quad (2.19)$$

The governing equation of the synthesis becomes

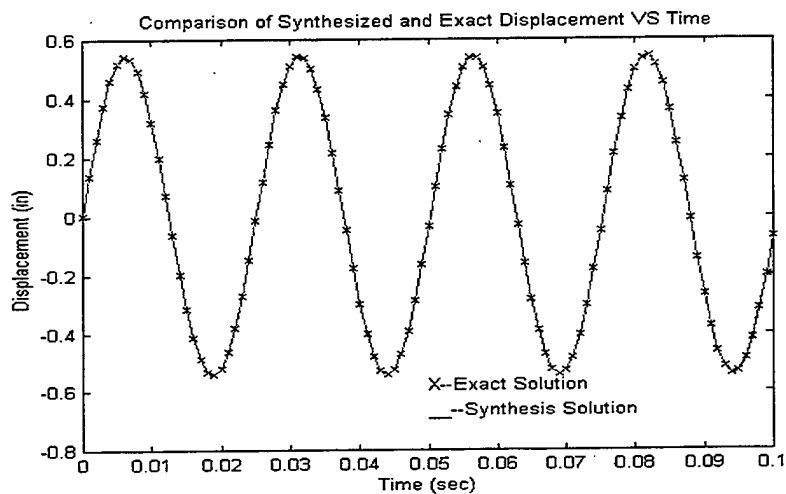
$$x^*(t) = x(t) - \frac{\Delta k}{\omega_n} \int_0^t x^*(\tau) \sin(\omega_n (t - \tau)) d\tau \quad (2.20)$$

Since this is a convolution integral, the transient response will be calculated at times  $t_i = i\Delta t$ ;  $i=0,1,2,3,\dots, n$ . Letting  $L_{ij} = \sin(\omega_n (i\Delta t - j\Delta t))$ , the integral in Equation (2.20) can be replaced by a trapezoidal rule

$$x^*(t_i) = x(t_i) - \frac{\Delta k}{\omega_n} \Delta t \left[ \frac{1}{2} L_{i0} x^*(0\Delta t) + \sum_{j=1}^{i-1} L_{ij} x^*(j\Delta t) + \frac{1}{2} L_{i0} x^*(i\Delta t) \right] \quad i = 0,1,2,\dots,n \quad (2.21)$$

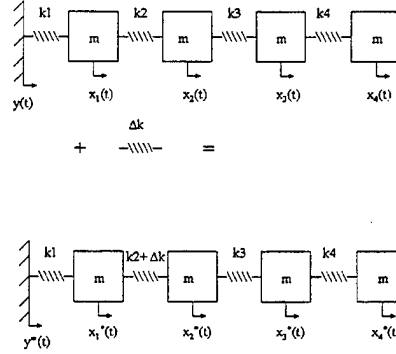
resulting in the lower triangular matrix as described in the previous section. The solution is obtained through forward substitution. Figure 2-4 provides a plot of the synthesized

transient response of the modified system and a comparison to the exact solution using a trapezoidal integration scheme. Note that, with respect to the resolution of the plot, the synthesized response exactly matches the exact response. A copy of the computer routine used is provided in Appendix A.



**Figure 2-4:** Comparison of Synthesized and Exact Displacement Response for Modified 1-DOF System

## Example 2: Multi-Degree-of-Freedom Stiffness Modification



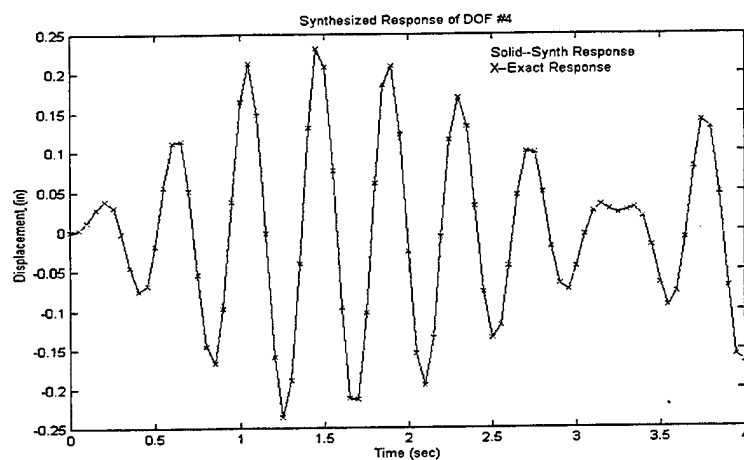
**Figure 2.5** Multi-DOF System with a Stiffness Modification at DOF #2

A somewhat more complex modification involves a multi-DOF system in which the modification (linear stiffness) and the DOF of interest are not connected, as shown in Figure 2.5. The spring-mass system is modified by increasing the stiffness of the second DOF by some value,  $\Delta k$ . With the second DOF modified, the transient response of the fourth DOF will be observed. Again, the synthesized transient response is generated by using the transient response of the original system in conjunction with its impulse response function (IRF). Given a harmonic forcing function of the form  $F(\sin\Omega t + \cos\Omega t)$  (where  $F$  is the amplitude and  $\Omega$  is the forcing frequency), the solution to this system's equation of motion requires the solution of four simultaneous equations. Since the system transient response is not so simply obtained as in Example 1, a computer is used to solve for the pre-modification transient response.

Applying Equation (2.18) once more, note that IRF calculation is simplified to :

$$h(t) = \sum_{p=1}^n \frac{\phi_i^p \phi_j^p}{\omega_{dp}} e^{-\zeta_p \omega_{np} t} \sin(\omega_{dp} t) \quad (2.28)$$

As in Example 1, the result is a convolution integral, which can be solved in a similar manner. Using a trapezoidal rule and forward substitution, the results are presented in Figure 2.6. Note that, as in Example 1, the synthesized solution is exact to within the resolution of the plot.



**Figure 2-6:** Synthesized Transient Response of DOF #4 to a Stiffness Modification at DOF #2



### III. NONLINEAR SYNTHESIS

The procedure outlined below was presented previously (Ref. [2]) for conducting a linear synthesis. The background and formulation for the nonlinear synthesis process are similar and are taken from References [2] and [4]. Rather than restating the formulation presented in Reference [2], a quick overview will be presented. Examples of both linear and nonlinear synthesis shall also be presented to show that the synthesis technique is similar for each case (albeit somewhat more computationally demanding in the nonlinear case). This synthesis formulation combines the computational efficiency of the time domain structural synthesis with the ability to install nonlinear modifications to the pre-modification structure. Through an iterative process, the exact solution is obtained quite rapidly.

#### A. GENERAL (LINEAR) SYNTHESIS FORMULATION

In this introductory section, all discussion of synthesis formulation will focus on the less complicated, linear problem. An iterative solution procedure is developed for the linear problem, which readily lends itself to use in the nonlinear problem. Thus, by understanding the formulation for the less complex, linear case, the reader is better able to understand the nonlinear problem. The section which follows deals specifically with the nonlinear synthesis formulation.

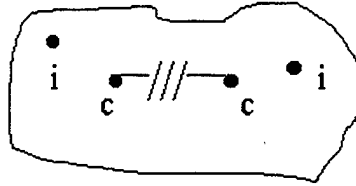
Consider a general NDOF system, as in Figure 3-1, with at least one rigid body mode, to which structural modifications are to be made. The first step in this process involves the assembly of the system stiffness, mass and damping matrices, either manually or through some finite element modeling subroutine. When the system matrices are assembled, the system natural and damped frequencies, as well as the system mass normalized modal matrix ( $[\Phi]$ ) are calculated. With this data and a given span of time over which the system's transient response is desired, the impulse response function



(IRF) matrix  $[h(t-\tau)]$  is determined for the pre-modification structure using the general IRF formula

$$h(t) = \sum_{p=1}^r \phi_i^p \phi_j^p t + \sum_{p=r+1}^n \frac{\phi_i^p \phi_j^p}{\omega_{dp}} e^{-\zeta_p \omega_{dp} t} \sin(\omega_{dp} t) \quad (3.1)$$

where all variables are defined as for Equation (2-18).



**Figure 3-1:** General NDOF System

Now that the system's pre-modification response function has been calculated, the system modification is "installed". Using the convolution integral equation (full synthesis equation), the modified system total dynamic response can be written as:

$$\{x(t)\}^* = \{x(t)\} + \int_0^t [h(t-\tau)] \{F(\tau)\}^* d\tau \quad (3.2)$$

where the force vector is now the modified force vector, which includes coupling forces between connection and base DOFs. Thus, the forces may be rewritten as:

$$\begin{aligned}
F_i &= F_i^{\text{ext}} \\
F_c &= F_c^{\text{ext}} + F_c^* = F_c^{\text{ext}} - (\Delta M_c \ddot{x}_c^* + \Delta C_c \dot{x}_c^* + \Delta K_c x_c^*) \\
F_b &= F_b^{\text{ext}} + F_b^* = F_b^{\text{ext}} - [(\Delta C_b (\dot{x}_b^* - \dot{y}) + \Delta K_b (x_b^* - y))]
\end{aligned} \tag{3-3}$$

where:

$F^*$ -the coupling forces due to structure modification

$[\Delta M]$ ,  $[\Delta C]$ ,  $[\Delta K]$ -the modification matrices

$x^*$ ,  $\dot{x}^*$ ,  $\ddot{x}^*$  -- generalized (synthesized) responses after modification

Combining equations (3-2) and (3-3) results in a nonstandard, nonhomogeneous VIDE of the second kind. The solution is obtained through the use of the trapezoidal quadrature rule as discussed in Section II, above. Also, by assuming no external excitations (i.e.  $\{F_e^{\text{ext}}\}=0$ ), the problem is reduced to the form

$$\begin{Bmatrix} x_i \\ x_c \\ x_b \end{Bmatrix}^* = - \begin{bmatrix} A_{ic} & A_{ib} \\ A_{cc} & A_{cb} \\ A_{bc} & A_{bb} \end{bmatrix} \left( \begin{Bmatrix} F_{\Delta M_c} \\ 0 \end{Bmatrix} + \begin{Bmatrix} F_{\Delta C_c} \\ F_{\Delta C_b} \end{Bmatrix} + \begin{Bmatrix} F_{\Delta K_c} \\ F_{\Delta K_b} \end{Bmatrix} \right) \tag{3-4}$$

where  $[A]$  is defined as follows:

$$[A_{ij}] = \Delta t \begin{bmatrix} \frac{1}{2}(h_{ij})_0 & 0 & 0 & 0 & 0 \\ \frac{1}{2}(h_{ij})_1 & \frac{1}{2}(h_{ij})_0 & 0 & 0 & 0 \\ \frac{1}{2}(h_{ij})_2 & (h_{ij})_1 & \frac{1}{2}(h_{ij})_0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ \frac{1}{2}(h_{ij})_n & (h_{ij})_{n-1} & (h_{ij})_{n-2} & \cdots & \frac{1}{2}(h_{ij})_0 \end{bmatrix} \tag{3-5}$$

In this quadrature matrix, the subscript n denotes the number of time steps. The coupling forces of equation (3-4) are defined as:

$$\begin{Bmatrix} F_{\Delta M_c} \\ 0 \end{Bmatrix} = \begin{bmatrix} [\Delta M_c] & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \ddot{x}_c^* \\ 0 \end{Bmatrix} \quad (3-6.a)$$

$$\begin{Bmatrix} F_{\Delta C_c} \\ F_{\Delta C_b} \end{Bmatrix} = \begin{bmatrix} [\Delta C_c] & 0 \\ 0 & [\Delta C_b] \end{bmatrix} \begin{Bmatrix} \dot{x}_c^* \\ \dot{x}_b^* - \dot{y} \end{Bmatrix} \quad (3-6.c)$$

$$\begin{Bmatrix} F_{\Delta K_c} \\ F_{\Delta K_b} \end{Bmatrix} = \begin{bmatrix} [\Delta K_c] & 0 \\ 0 & [\Delta K_b] \end{bmatrix} \begin{Bmatrix} x_c^* \\ x_b^* - y \end{Bmatrix} \quad (3-6.c)$$

The solution to this problem is best obtained through an iterative procedure due to the large size of [A]. The iteration process followed is outlined below:

- 1) Assume the force vector  $\{F_z\}$
- 2) Calculate the response vector  $\{x_e\}^*$
- 3) Use  $\{x_e\}^*$  to calculate  $\{\dot{x}_e\}^*$ ,  $\{\ddot{x}_e\}^*$ , and a new  $\{F_z\}$
- 4) Use the new  $\{F_z\}$  and calculate a new  $\{x_e\}^*$
- 5) Repeat steps 3 & 4 until  $\{x_e\}_{new}^* - \{x_e\}_{old}^* \leq \text{convergence tolerance}$

Through the use of the trapezoidal quadrature and the iterative methods, approximations have now been introduced to the time domain synthesis method (which is an exact formulation in itself). Convergence and stability become issues of concern with the iterative procedure. Smaller time step size ( $\Delta t$ ) will lead to an increase in stability and a more rapid convergence by keeping the norm of [A] small. With smaller time step size comes the need for larger computer memory requirements for a given time interval of interest. There may be limitations on the span of time which may be "observed" by this analysis, dependent on the computer system used, due to memory capabilities.

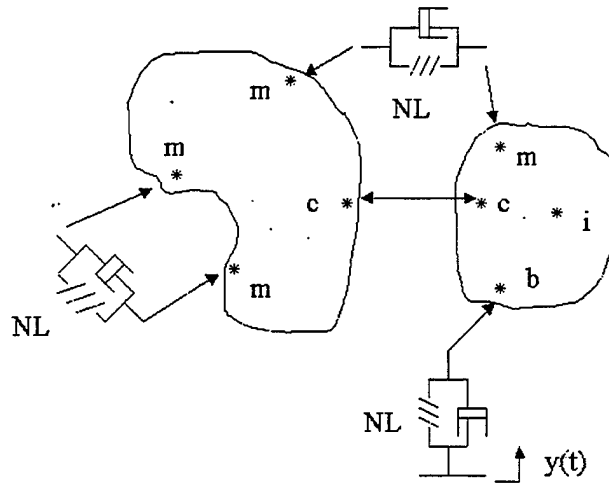
## B. NONLINEAR SYNTHESIS FORMULATION

Given that an entire model is rendered nonlinear by the inclusion of a single nonlinear element, traditional direct integration solutions become computationally expensive. By isolating the nonlinearities from the bulk model the majority of the model gains the benefits of linearity, simplifying calculations. Also, by retaining only those degrees-of-freedom (DOF) of interest to the analysis, a more efficient response analysis is developed. This formulation allows for repeated, rapid nonlinear transient analysis for large, linear structural FE models, which can have localized, generally nonlinear components.

Using only those DOF of interest in physical (non-modal) coordinates, a transient analysis of an arbitrary structure may be conducted which is independent of model size. The retained DOF must include, as a minimum, those DOF associated with the nonlinear elements, and those DOF of which the synthesized transient response is desired. Thus, it is possible to solve for the transient response of an arbitrarily large model using an arbitrarily small number of DOF (limited only by the number of nonlinearities and points of interest). This feature has been demonstrated for both the linear and nonlinear formulation References [2] and [4], respectively.

As mentioned previously, it is desirable to isolate the nonlinear elements from the bulk, linear model. In order to achieve this, the structure must be divided into a number of substructures, Figure 3-2. By dividing the structure in this way, linear portions of the model may remain linear. For those linear substructures subjected to excitation, the baseline transient response is calculated. Any substructure not subjected to this excitation only requires that the impulse response functions be generated. This is known as a substructure coupling approach. With the nonlinear elements isolated, this method provides computational advantage by exploiting inherent physical boundaries in the problem and maintaining substructure linearity. The synthesis is used to connect the linear substructures through the nonlinear elements, and to calculate the combined system nonlinear transient response. Once the nonlinear elements are connected, the entire system is rendered nonlinear and the nonlinear transient response is described exactly by the governing equations of synthesis. The iterative method introduced in the previous

section on general synthesis is applied to the nonlinear synthesis problem with similar results. One important modification, which simplifies calculations and reduces computer memory requirements, has been made in the solution process. Rather than generating the quadrature matrix at each iteration, the appropriate IRF for the point of interest is convolved with the associated force vector  $\{F_z\}$  using the MATLAB CONV command (Reference [5]). For large systems, this results in a considerable reduction in the number of calculations required to complete the synthesis (and thus, improved timesavings). This savings in time and memory requirements allows for increased stability and convergence capability by permitting even smaller time step sizes for the same memory requirements as the general synthesis procedure described in the previous section.



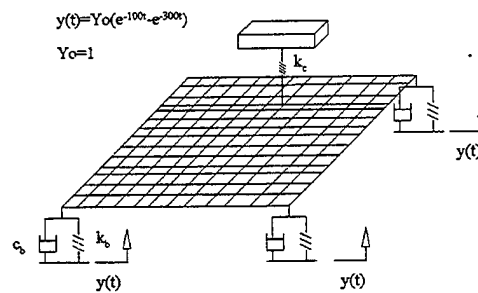
**Figure 3-2:** Substructure Coupling/Modification for Nonlinear Synthesis

## C. EXAMPLES

The validity/accuracy of the time domain synthesis technique has already been shown in Refs. [3 and 4]. The following examples are provided simply to demonstrate the ability of this method to solve for large, linear and locally nonlinear models with the same algorithm and equivalent performance.

### Examples 1 and 2: Mass-Plate System

Consider the following mass-plate system, presented as Figure 3-3:

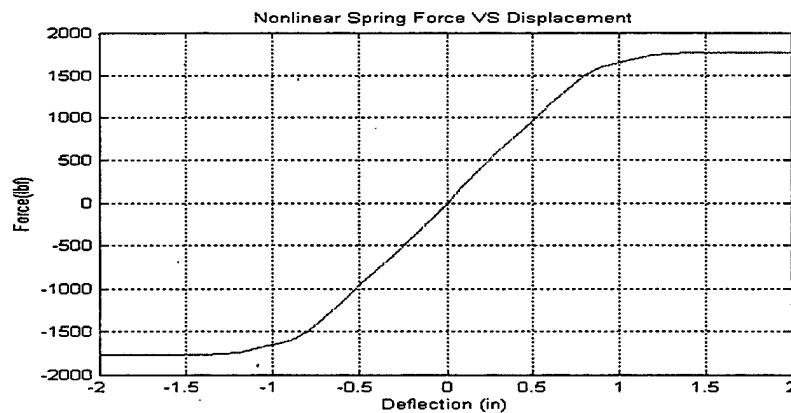


**Figure 3-3: Mass-Plate System Experiencing Base Excitation**

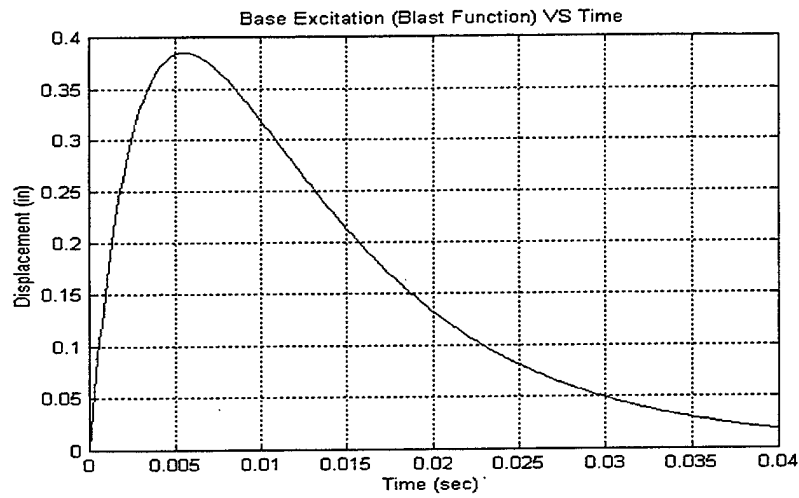
This mass-plate system is the same as used in Ref. [4]. The stiffeners and dampers are added as modifications to the baseline system. The same system is used in both Examples 1 and 2. However, in Example 1, the stiffeners added possess a linear force-displacement characteristic of 500 lb<sub>f</sub>/in, while the stiffeners in Example 2 possess the nonlinear force-displacement characteristic curve shown in Figure 3-4. In both examples, the dampers possess a linear damping characteristic of 2 lb<sub>f</sub>-s/in. The stiffeners used in Example 2 are representative, but not an exact reproduction, of the starting point to be used for the optimization routine.

The original structure is represented by the plate, which has spring-to-base attachments at each corner, and the attached computer which is located off-center, above, and connect to the plate with a linear (1000 lb/in) spring. (Note that base isolators are not installed in the pre-synthesis model, making it a free-free structure.) The plate-computer system is approximately 51,500 DOF. The computer is modeled as a single lumped mass with a single DOF in translation only. The base excitation,  $y(t)$ , is a blast function and is presented as Figure 3-5.

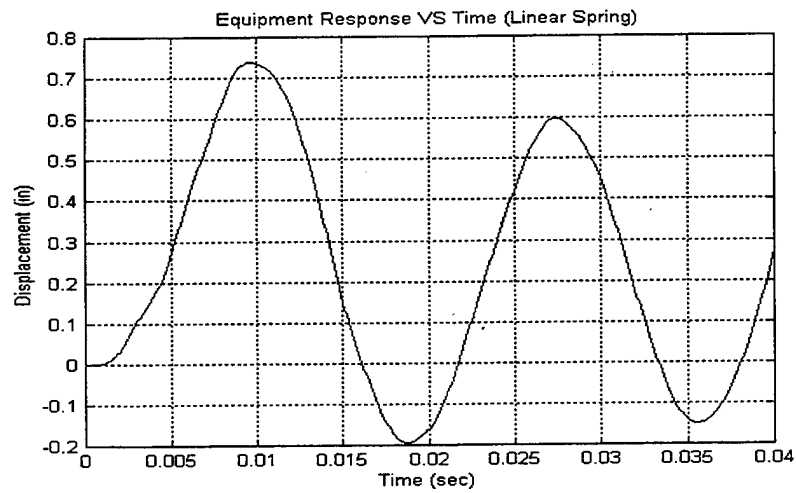
Figure 3-6 is a plot of the transient response of the computer to a blast function with linear isolators installed. Figure 3-7 is a plot of the transient response of the computer using the same base excitation with nonlinear stiffeners and linear dampers installed. The synthesis algorithm for both analyses is the same and is provided in Appendix B. Note that the only change made to the algorithm involves replacing the linear isolator with the nonlinear isolator. The synthesis performed for the linear isolator required 22.09 sec, while the synthesis for the nonlinear isolator required 50.84 sec.



**Figure 3-4:** Nonlinear Spring Force-Displacement Characteristic

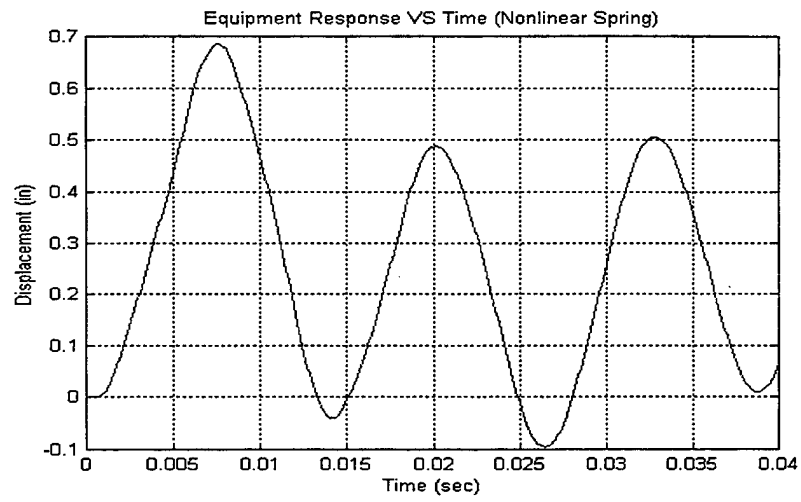


**Figure 3-5: Base Excitation  $Y(t)$  (Blast Function)**



**Figure 3-6: Transient Response of Computer with Linear Spring and Damper Modification**





**Figure 3-7:** Transient Response of Computer with Nonlinear Spring and Linear Damper Modification

## IV. OPTIMIZATION

The background information presented here is an overview of what is presented in reference [6]. With the introduction of Commercial-Off-the-Shelf (COTS) technology into military platforms comes the need to ensure that such equipment is properly isolated from shock and vibration. Since COTS are not designed to meet military specifications with regard to shock survivability, shock isolation systems must be developed which ensures equipment survivability in a combat environment. However, to physically test each equipment configuration until a suitable mount is found would become prohibitively expensive. A more cost-effective approach involves modeling equipment and mounting systems on a computer. Computer simulations could then be conducted to determine the feasibility of different mounting configurations.

The problem is still not so simple as it may appear. Design of a shock only or vibration only - isolation system may be relatively easy. However, when trying to isolate both shock and vibration, the optimal design for one is in conflict with the other (References [7] and [8]). This seriously complicates the optimal design of a shock and vibration isolation system. Thus, the objective of the optimization becomes the minimization of some aspect of the system response while also constraining other response characteristics within some reasonable bounds.

Even this does not fully describe the complexity of the problem presented. Since the system to be isolated is typically large, the calculations alone are demanding. Add to this the need to explore variations in a large number of design variables and the optimization becomes more cumbersome. Finally, given that an entire model is rendered nonlinear by the inclusion of a single nonlinear element, traditional direct integration solutions of a single configuration become computationally expensive. Combine this with an optimization routine requiring numerous configuration tests, and the optimization of nonlinear isolators becomes, for all practical purposes, prohibitively expensive. However, by further developing the work presented in Refs. [3 and 4], an efficient

process for determining the optimal nonlinear isolation characteristics for large, locally nonlinear system responses can be obtained.

By isolating the nonlinearities from the bulk model the majority of the model gains the benefits of linearity, simplifying calculations. Also, by retaining only those degrees-of-freedom (DOF) of interest to the analysis, a more efficient response analysis and optimization routine is developed. This formulation allows for repeated, rapid nonlinear transient analysis for large, linear structural FE models, which can have localized, generally nonlinear components.

## **A. OPTIMIZATION ROUTINE FORMULATION**

In the formulation of an optimization routine, several specific tasks must be accomplished to define the optimization space. First, an objective function must be defined which specifies the quantity to be minimized or maximized. Second, design variables (those quantities to be varied in the optimization) must be determined and initialized. Finally, the problem must be constrained in some way, and constraint functions, which are themselves functions of the design variables, must be developed. These constraints help limit the size of the optimization space, allow for the imposition of physical restrictions, and when properly chosen, can reduce the optimization search time. Three types of constraints may exist in an optimization problem. First, an inequality constraint is one in which the value calculated from the constraint function is less than or equal to some prescribed limit. Second, an equality constraint is one in which the constraint function value is equal to some prescribed value. Finally, a side constraint is one where some upper and lower limits are placed on the design variables.

For the design of an isolation system, the minimization of the system's response due to some excitation may be the ultimate objective. The minimization of response history away from equilibrium, the absolute or relative displacement, or perhaps the absolute acceleration of the system may be important. Any one of these responses may become the objective function while the other responses may be used as constraints on the system. While minimizing the maximum acceleration experienced by the equipment

may be the primary concern, returning the system to equilibrium as quickly as possible, and the absolute displacement or the stretching of the isolators (relative displacement) must fall within some limits to avoid undesirable damage. To achieve the objective, physical parameters such as isolator stiffness and damping might be altered. Thus, as in Reference [2] the following optimization problem is generated:

**Minimize (Objective Function):**

Maximum system acceleration due to base excitation

**Subject to (Constraints):**

Maximum dynamic isolator stretch/compression  $\leq$  limit

Motion away from equilibrium  $\leq$  limit

Maximum equipment acceleration  $\leq$  limit

Min and max changes in isolator stiffness  $\leq$  limit

Min and max changes in isolator damping  $\leq$  limit

**Design Variables:**

Changes in the isolator stiffness characteristics

Changes in the isolator damping characteristics

# **1. Calculation of Objective Function**

The objective function is based on the system's (computer) maximum acceleration due to some base excitation, which may be any random vibration or shock input. The displacement response is calculated first, then a finite difference scheme (which accounts for beginning (forward scheme) and ending (backward scheme) history points) can be used to calculate the acceleration response. From this acceleration time history, a maximum of the absolute accelerations experienced is readily obtained. By minimizing the accelerations experienced by the computer, the potential for damage is minimized. The objective function becomes

$$f = \max \left( \left| \frac{d^2 x(t)}{dt^2} \right| \right) \quad (4-1)$$

For each iteration of the optimization process Equation 4-1 must be evaluated. This is accomplished by first calculating the system (computer) response through the use of the Time Domain Structural Synthesis for each variation of the optimization routine. The transient response of the computer is generated then used to evaluate the objective function.

## **2. Calculation of Constraints**

The first two constraints noted are of particular concern in the formulation of this problem. Possibly the most important constraint is held on the maximum dynamic isolator stretch/compression. While the isolator stiffness (and therefore resistance) increases with increasing stretching or compression, there are physical limitations to the amount of stretching or compression an isolator may sustain without damaging or altering its force-displacement characteristic curve. Therefore, it is desirable to constrain the relative displacement between the plate and the base to some reasonable value within equipment specifications. In order to calculate this constraint, the transient response of both the base excitation and the structure (plate) at the points of isolation must be known (or calculated) for the time history of interest. Since the base excitation is prescribed, only the structure's displacement time history need be determined. The time domain structural synthesis (nonlinear) technique is applied to calculate this displacement time history.

The second constraint is a limit on the time required for the system to return to equilibrium after some base excitation. Since the primary concern, for this problem, is the isolation of equipment in a combat environment, the assumption is made that it is desirable to stabilize the equipment as quickly as possible without introducing large accelerations. This constraint, if overly restrictive, can conflict with the objective function. In order to return the system to equilibrium rapidly, larger accelerations will be developed than if the system is brought to equilibrium more gradually. Thus, there is a

trade-off between ensuring the equipment is protected from violent accelerations and returning the system to equilibrium rapidly. Thus, with the objective function taking priority, this constraint is simply a limitation on the amount of time the system requires to return to some acceptable level of stability. Note, that in many environments (seismic excitation of a structure, for example) a rapid return to equilibrium may not have any significance, and would not be considered a constraint.

The third constraint helps drive the optimization routine past any potential local minima which would produce an undesirable level of acceleration. This constraint limits the maximum acceleration that may be experienced by the equipment and ensures that local minima which do not meet this criteria are discarded as invalid solutions.

The last two constraints noted are simply limitations on the degree to which values of the coefficients of the characteristic force-displacement curve of the isolator may change for each optimization iteration. These constraints can be controlled by pre-defining design variable change limits for the optimization function.

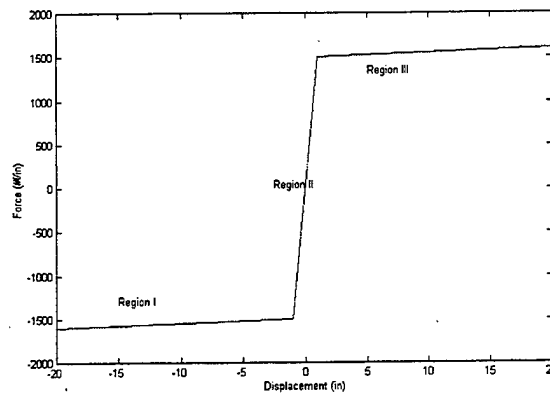
### **3. Design Variables**

Isolator stiffness and damping characteristics are changed during this procedure to optimize the response of the equipment. However, there are limitations to the amount of change these characteristic curves may undergo. The most obvious limitation is that the lower bound for the isolator stiffness characteristic is not equal to the zero. Use of this stiffness would be equivalent to the complete removal of the isolator. The upper bound of isolator stiffness and damping is governed simply by what is physically reasonable. One additional limitation is introduced with the nonlinear problem. As the characteristic force-displacement curve turns to change slope with increasing displacement/compression (Regions I and III in Figure (4-1) below), the new slope must be greater than or equal to zero. A negative slope in these regions will result in an instability in the synthesis procedure, which may results in an inability of the synthesis method to converge. A negative slope in these regions also indicates a system which is physically infeasible. In the case of a negative slope, spring forces would not oppose

compressions/expansions, but rather be sympathetic to them. This problem can be avoided by proper choice of the lower and upper bounds set on the design variables which govern slope in regions I and III, such that a negative slope is not possible without violating boundary conditions. By properly choosing two points (or two slopes) on a force-displacement space and reflecting them across the displacement axis, a force-displacement characteristic curve is generated. Allowing the optimizer to alter the points/slopes chosen (such that the constraints prevent a negative slope in Regions I or III as mentioned above), a force-displacement characteristic of the optimal isolators can be generated.

## **B. OPTIMIZATION COMPUTER CODE**

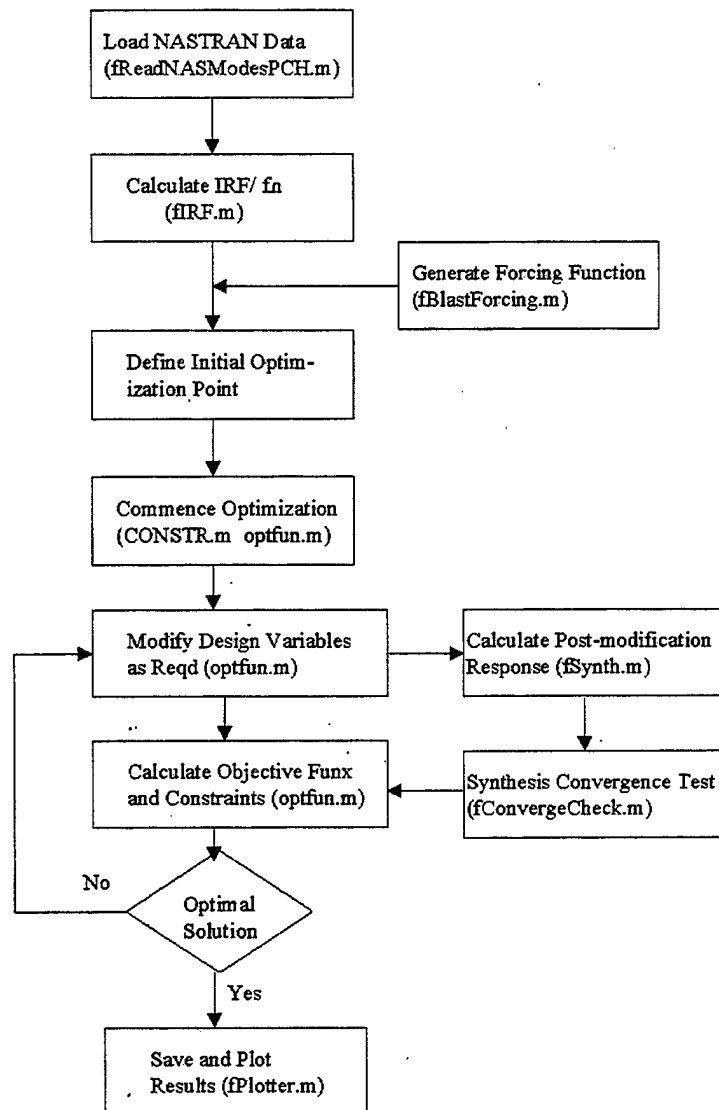
The optimization tool used to solve this problem, as in Ref. [3], is the MATLAB optimization toolbox 1.0d. Since the problem posed is nonlinear, constrained, and multivariable, the function CONSTR.m is used since it is specifically designed to solve such problems. CONSTR uses the Sequential Quadratic Programming (SQP) method in order to solve the problem (Reference [9]). The SQP method is used to determine the search directions for the design variables for each iteration (Refs. [5] & [9]). As with many nonlinear optimization computer routines, this method has the important limitation that the optimal solution found might only be a local solution. Thus, improvement might be made upon the solution obtained. This highlights the need for the user to have a good physical understanding of the problem so that an appropriate starting point is chosen. Another difficulty, which might be encountered, occurs when the problem posed is infeasible. In this case, CONSTR.m attempts to minimize the maximum constraint value. This can lead to the search pattern extending outside the defined optimization space, the user defined iteration limit being exceeded, and/or no optimal solution being found.



**Figure 4-1: Sample Nonlinear Force-Displacement Curve**

The computer code used in this optimization routine follows the flow path presented in Figure 4-2. Model characteristics are loaded from a NASTRAN data file through the use of the function `fReadNASModesPCH.m`. Model natural frequencies  $\{\omega_n\}$ , Impulse Response Functions (IRFs), and the forcing function are generated prior to starting the optimization routine. A starting point and upper and lower bounds for the optimization design variables are defined and `CONSTR` is invoked. For each iteration of `CONSTR`, the optimization function evaluates the post-modification model (based on design variables), calculates the value of the objective function, and verifies that constraints have not been violated. The routine remains within the optimizer and continues to modify the model until an optimal solution is located. A copy of the computer code used is enclosed as Appendix C.

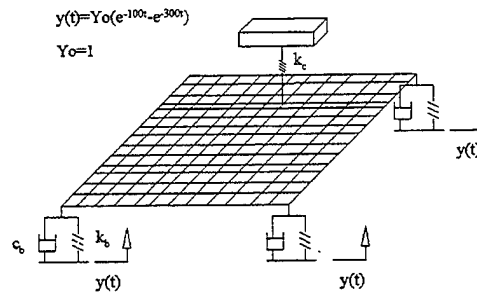




**Figure 4-2: Optimization Routine Flowpath**

### C. EXAMPLE

Consider the following computer-plate system:



**Figure 4-3:** Plate-Mass System Experiencing Base Excitation

The original structure is represented by the plate, which has spring-to-base attachments at each corner, and the attached computer which is located off-center, above, and connect to the plate with a linear (1000 lb/in) spring. (Note that base isolators are not actually installed in the pre-synthesis model.) The plate-computer system is approximately 51,500 DOF. The computer is modeled as a single lumped mass with a single DOF in translation only. The base excitation,  $y(t)$ , is a blast function identical to that presented previously in Figure 3-5. All solutions are calculated using a MICRON 266 MHz Millennium computer.

## 1. Problem Formulation

The following is the formal problem statement for optimization:

$$\begin{aligned}
 \text{Minimize: } & F(\Delta k_b, \Delta k_c, \Delta c_b, \Delta c_c) = \max \left( \left| \frac{d^2 x(t)}{dt^2} \right| \right) \\
 \text{Subject to: } & \left\{ \begin{aligned}
 g_1 &= \frac{k_{p\_stretch}}{\max k_{p\_stretch}} - 1 \leq 0 & 0.5 \leq x_{b1} \leq 1.5 \\
 g_2 &= \frac{k_{c\_stretch}}{\max k_{c\_stretch}} - 1 \leq 0 & 7.21 \leq x_{b2} \leq 9.65 \\
 g_3 &= \frac{\frac{1}{2} \int_0^t |x^*(\tau)|^2 d\tau}{\max \left( \frac{1}{2} \int_0^t |x^*(\tau)|^2 d\tau \right)} - 1 \leq 0 & 450 \leq k_{b1} \leq 700 \\
 g_4 &= \frac{x_{c\_accel}}{\max x_{c\_accel}} - 1 \leq 0 & 701 \leq k_{b2} \leq 710 \\
 g_5 &= \frac{x_{c\_dynam}}{\max x_{c\_dynam}} - 1 \leq 0 & 0.0 \leq c_b \leq 5.0
 \end{aligned} \right.
 \end{aligned}$$

where:  $y_c(t)^*$  = translational motion time history for the computer

$k_{p\_stretch}$  = isolator stretch between base and plate

$k_{c\_stretch}$  = isolator stretch between plate and computer

$\max k_{p\_stretch}$  = max. allowable isolator stretch between base and plate

$\max k_{c\_stretch}$  = max. allowable isolator stretch between plate and computer

$x_{c\_accel}$  = absolute acceleration of computer due to shock [in/s<sup>2</sup>]

$\max x_{c\_accel}$  = max. allowable absolute acceleration of computer [in/s<sup>2</sup>]

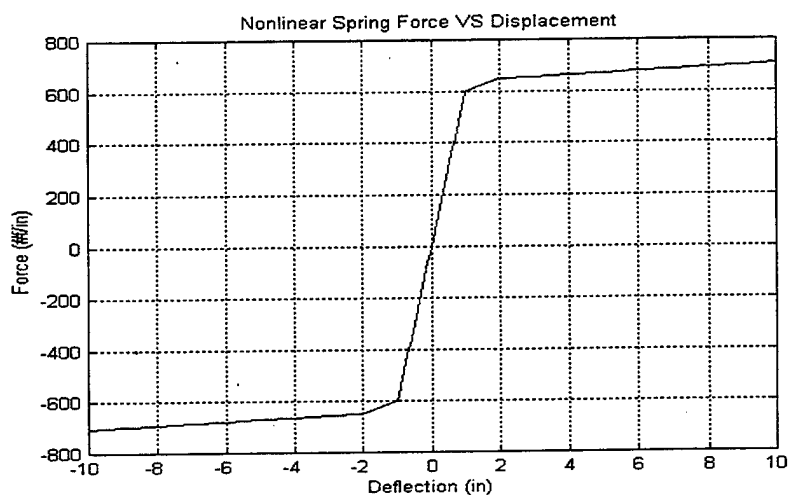
$x_{c\_dynam}$  = absolute dynamic displacement of computer due to shock

$\max x_{c\_dynam}$  = max. allowable dynamic displacement of computer

$x_{b1}, x_{b2}, k_{b1}, k_{b2}$  = plate isolator design variables

$c_b$  = plate damper design variable

The following information is provided as a starting point for the optimization search routine. These points produce a rough approximation of the force-displacement characteristic curve presented in Figure 4-4. Table 4-1 provides these points along with upper and lower limits for more reasonable isolator profiles.



**Figure 4-4:** Starting Point Force-Displacement Characteristic

	$x_{b1}$	$x_{b2}$	$k_{b1}$	$k_{b2}$	$c_{b1}$
<b>Initial</b>	1.0	9.65	600	705	0.5
<b>Lower</b>	0.5	7.21	450	701	0.0
<b>Upper</b>	1.50	10.00	700	710	5.0

**Table 4-1.** Optimization Start Point Inputs

The following inequality constraints are also placed on the optimization routine:

$$\max k_p\_stretch \leq 0.50 \text{ inch}$$

$$\max k_c\_stretch \leq 0.50 \text{ inch}$$

$$\max x_c\_dynam \leq 1.00 \text{ inch}$$

$$\max x_c\_accel \leq 15 \text{ g's}$$

As mentioned above, the proper choice of the starting point is of particular importance. Since the optimization routine solution may result in a local minimum, the user must use his/her knowledge of the system to choose an appropriate starting point.

## 2. Optimization Results

This example is executed using the following main programs and their associated subroutines:

### Main Programs:

1. fMainOpt

2. Optfun

3. Synth2opt

### Subroutines Called:

fReadNASModesPCH

fIRF

fBlastForcing

fddot

trapz

fSymmetricStore

conv

fConvergeCheck

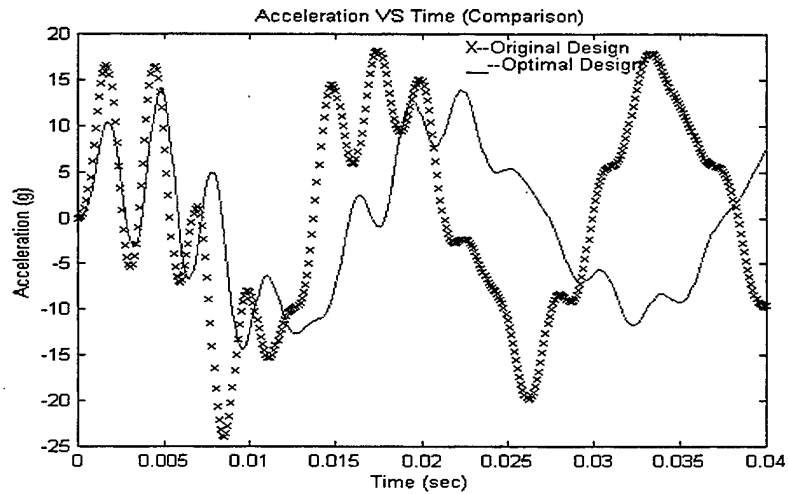
fExceed\_Iter

When the optimization routine is complete, the following information and plots are generated which help the user understand the optimization process. Table 4-2 is the display presented on the MATLAB Command Window, which provides information regarding the progress of the optimization routine. The first plot, Figure 4-5, is a comparison of the starting point acceleration response to the "optimal design" acceleration response of the computer after the modifications have been installed. The starting point design experiences a maximum acceleration of 23.4 g's while the optimal design experiences a maximum acceleration of 14.3 g's.

<u>f-COUNT</u>	<u>FUNCTION</u>	<u>MAX{g}</u>	<u>STEP</u>	<u>Procedures</u>
6	23.9362	7.93622	1	
12	14.7974	-0.12069	1	
18	14.7845	-0.120815	1	Hessian modified
24	14.3508	-0.125029	1	
30	14.3472	-0.125064	1	Hessian modified
31	14.3472	-0.125064	1	Hessian modified

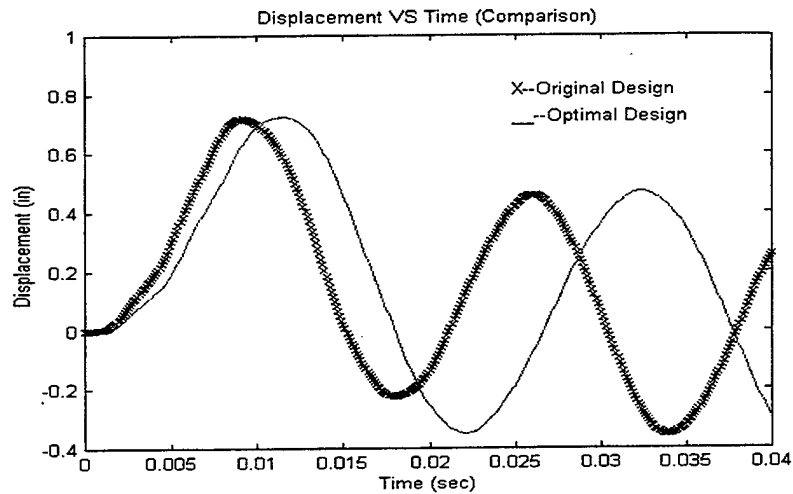
Optimization Converged Successfully

**Table 4-2:** Optimization Routine MATLAB Generated Output



**Figure 4-5:** Comparison of Acceleration VS Time Between Starting Point and Optimal Design Modifications

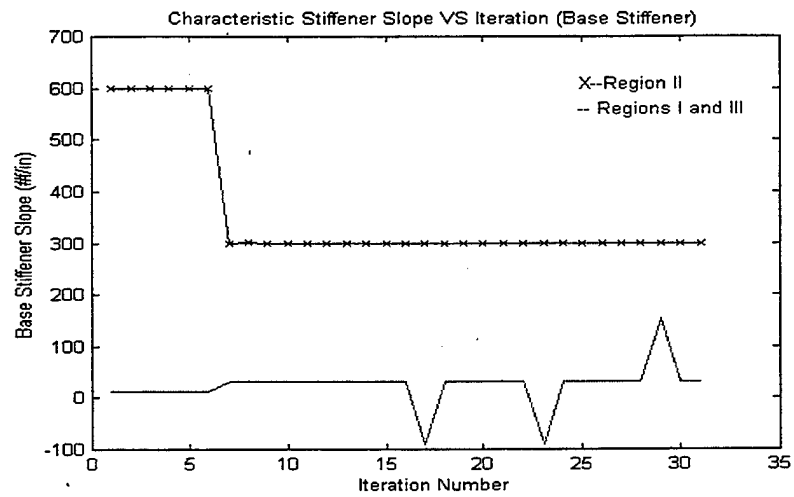
Figure 4-6 is a comparison plot of the starting point displacement response to the optimal design displacement response of the computer. Note that while maximum displacements have not been altered to any great degree, there is a more gradual change of displacement with time in the optimal design, leading to lower accelerations. Effectively, the natural frequencies of the system have been lowered.



**Figure 4-6:** Comparison of Displacement VS Time Between Starting Point and Optimal Design Modifications

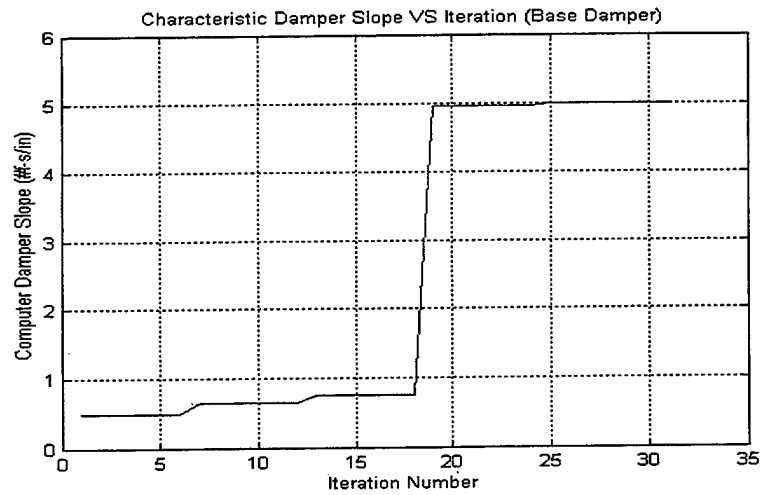
Figure 4-7 is a plot of the slopes that characterize the nonlinear spring force-displacement profile at each iteration of the optimization routine. Note that the slope of regions I and III become negative in iterations 17 and 23. These points violate a constraint, are not valid solutions, and are thus rejected by the optimization routine.





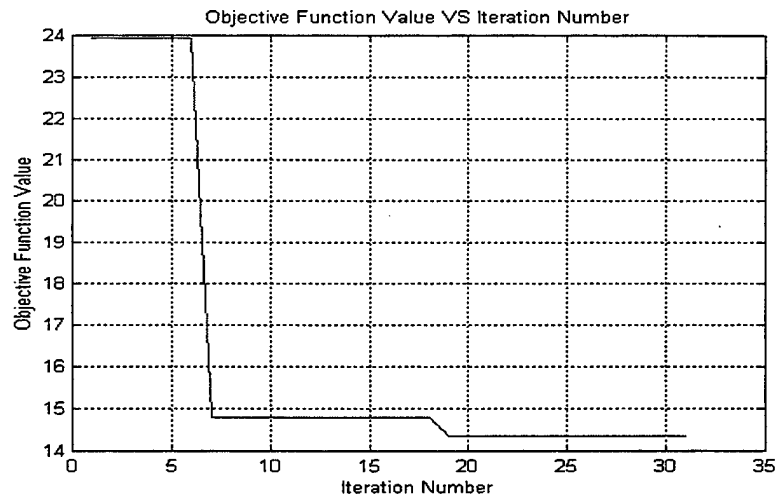
**Figure 4-7: Change in Stiffener Slope (Base Isolator) VS Iteration Number**

Figure 4-8 is a plot of the installed damper characteristic slope versus iteration number.



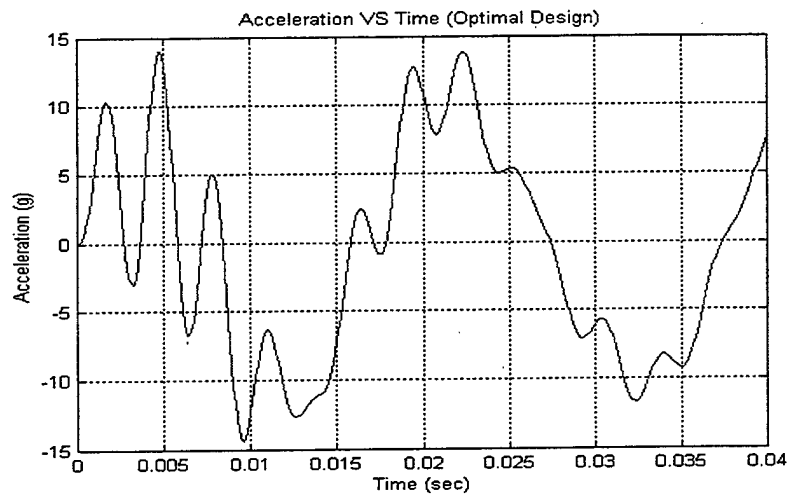
**Figure 4-8: Base Damper Characteristic Slope VS Iteration Number**

Figure 4-9 is a plot of the value of the objective function versus the number of iterations.

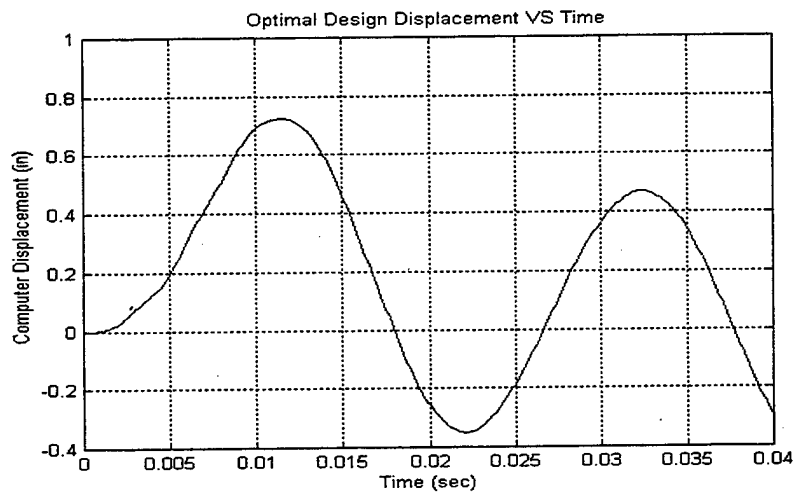


**Figure 4-9:** Objective Function Value VS Iteration Number

Figures 4-10 and 4-11 present the optimal design acceleration and displacement response time histories, respectively.



**Figure 4-10:** Optimal Design Acceleration VS Time Response



**Figure 4-11:** Optimal Design Displacement VS Time Response

Table 4-2, and Figures 4-5 and 4-6 give a clear picture of how modifications to the structure can have such a large impact on system response. Since the objective function is to minimize the maximum of the absolute of the acceleration, it is readily visible, in Figure 4-5, how much the design has improved from starting point to "optimal" point. This reduction in the value of the objective function is even more apparent in Figure 4-9. Starting with an initial value of 23.4 g's (with the starting point design variables), the optimization routine quickly reduces the value of the objective function by nearly 40%, to 14.3 g's.

Overall, this optimization required 31 iterations to determine optimal values for the design variables to minimize the objective function and meet the constraint conditions. This entire process required only 15 minutes and 47 seconds (for an average of 30.5 seconds per iteration). Possibly somewhat more impressive is the fact that an optimal solution was found so rapidly given that 5 design variables needed to be manipulated. Thus, this optimization routine calculated the transient response of a 51,500 DOF structure, determined the maximum acceleration experienced for each modification and verified that constraints were met, given 5 variables to be manipulated, and still produced a valid solution in roughly 16 minutes.

To verify the validity of the solution, several starting points close to the "optimal" were tested. Each time the same final solution was obtained in less than 10 minutes. Table 4-3 presents the starting points and time requirements for the optimization to return a solution. In each case, the optimal values for design variables are identical to those presented in Table 4-2 and are not reproduced here.

RUN #	Starting Point Values					Time Req'd (sec)
	$x_{b1}$	$x_{b2}$	$k_{b1}$	$k_{b2}$	$c_{b1}$	
1	1.0	9.65	600	705	0.5	947.31
2	1.45	9.65	500	705	4.5	531.14
3	1.35	9.55	490	704	4.9	375.90

**Table 4-3:** Comparison of Starting Point Values VS Time Required



## V. CONCLUSIONS/RECOMMENDATIONS

This thesis was undertaken to investigate the feasibility of applying an optimization routine to the isolation (from shock and/or vibration) of large, locally nonlinear structures. Traditional FEA methods require the user to rebuild the model for each modification and the solution of the entire model. However, using the Time-Domain Structural Synthesis method, the pre-modification model is solved a single time and the solution of the transient response of an arbitrarily small number of DOF is required, resulting in exceptional timesavings. The work presented in References [1], [2], and [4] is the foundation for this thesis and has been presented, in part, as background. An important modification to the methods presented in these references, which improves time and memory requirements, was used in this thesis. While this modification did not significantly alter the procedures used, it removed the need to generate the quadrature matrix at each iteration and resulted in a more rapid synthesis solution. This improvement was incorporated in the synthesis technique and coupled with the optimization routine to determine the optimal configuration of an isolation system given a pre-modification model.

### A. CONCLUSIONS

The following conclusions can be drawn from the analyses presented:

1. The time-domain structural synthesis method is an efficient and accurate method of calculating the transient response of a structure following modification. The position and degree of modification is unrestricted by the method. Since numerical techniques are used to produce a solution, some error is introduced, through approximation, to the final solution. Errors introduced by numerical solution are limited by decreasing the time step size used. By generating only those values used for each convolution, at each



iteration, time step size may be further reduced since less memory is required (compared to generating the quadrature matrix for each iteration).

2. Use of the time-domain synthesis method is integral in the optimal design of a nonlinear isolation system for a large structural system. Having demonstrated order of magnitude (or better, Reference [4]) improvements over traditional FE methods, in calculation time, the result is more rapid solution of the optimization problem. Through the use of this technique an optimal isolation system for a 51,500 DOF structure was generated in less than 16 minutes (15 minutes 47 seconds), using nonlinear stiffeners and linear dampers as design variables. After 31 different configuration tests (averaging 31 seconds per configuration) an optimal solution was presented. As a comparison, tests in Reference [4], using the same model, showed that a direct FE solution required 43 minutes 41 seconds to solve for the transient response of a single configuration. Further, the older synthesis routine used in Reference [4] (in which the quadrature matrix is generated for each iteration) required 4 minutes 15 seconds to produce a transient response of a single configuration of the same model used in this thesis. Recognizing that the processor used in Reference [4] is different than the one used here, a better comparison of time requirements must be based on a comparison of the processor speeds used in each test. Based on the different processor speeds used, there is still better than an order of magnitude reduction in calculation time over the direct FE method, and an 80% reduction in calculation time over the older synthesis routine .

## **B. RECOMMENDATIONS**

While conducting this study, several important areas for improvement were noted. Recommendations to address these potential improvements include:

- Develop a more intricate nonlinear stiffener generator in the optimization routine and a more robust decision algorithm for the synthesis to determine stiffener force/deflection as the synthesis converges.
- Develop a more complex and realistic model of the damping portion of the isolator.



## **APPENDIX A. TIME-DOMAIN SYNTHESIS COMPUTER CODES**

### **FOR EXAMPLES 2-1 and 2-2**

The following is a brief description of the MATLAB computer codes used to perform the calculations necessary for Examples 2-1 and 2-2. Both examples shown present a linear structural modification.

- VIBE1DOF.m – performs structural modification on a 1-DOF spring-mass system. Using time-domain structural synthesis to calculate the synthesized transient response of the post-modification system to a base excitation.
- VIBEMDOF.m – performs structural modification on an M-DOF spring mass system experiencing base excitation. Allows the user to choose the location of the modification and the DOF where the transient response is desired. This program also uses the time-domain synthesis technique to calculate the post-modification transient response.

VIDE1DOF.m

```

%*****
% Define constants
%*****

kspring=100.0;           % spring stiffness #/in
mass=100/386.4;          % mass (#m)
dkspring=0.4*kspring;    % spring modification
W=250;                   % Forcing Frequency
f0=1.1;                  % Forcing Amplitude
zeta=0.05;                % Damping Factor

%*****
% Set time step information
%*****

ti=0.0;                  %Start time
tf=.10;                  %Stop time
dt=1e-3;                 %Time step size
numstep=(tf-ti)/dt;      %Number of time steps

%*****
% Calculate several constants
%*****

wn=sqrt(kspring/mass);    % Natural Freq. (rad/sec)
wd=wn*sqrt(1-zeta^2);    % Damped Nat Freq
c1=zeta*wn;               % Exact Soln Consts
c2=c1^2;
c3=c2+wd^2+2*wd*W+W^2;
c4=c2+wd^2-2*wd*W+W^2;
c5=W^2-c2-wd^2;
c6=W^2+c2-wd^2;

%*****
% Initialize Kernel Matrix and begin calculations
%*****

K=zeros(numstep+1,numstep+1);
K(1,1)=1.0;
C=dt;

```

```

for i=2:numstep+1
    t=(i-1)*dt;

    for j=1:i
        tau=(j-1)*dt;
        if i>j          %Lower Triangular
            if j==1      %First Column

                kernel=(1/wd)*exp(-zeta*wn*(t-tau))*sin(wd*(t-tau));
                K(i,j)=C/2.*kernel;

            else          %2nd + columns
                kernel=(1/wd)*exp(-zeta*wn*(t-tau))*sin(wd*(t-tau));
                K(i,j)=C*kernel;

            end
        elseif i==j      %Main Diagonal

            kernel=(1/wd)*exp(-zeta*wn*(t-tau))*sin(wd*(t-tau));
            K(i,j)=1.+C/2*kernel;

        end
    end
end

%*****
% Calculation and assembly of the x(t) vector
%*****

tp=ti:dt:tf;
tp=tp';

p1=f0.*(c5.*sin(tp.*W)+2.*c1.*W.*cos(tp.*W))/(c3+c4);
p2=f0.*W.*exp(tp.*(-c1)).*(c6.*sin(tp.*wd)...
    +2.*c1.*wd.*cos(tp.*wd))./(wd*c3*c4);
x=p1+p2;

xstar=K\x;
maxK=max(eig(K))

wnmod=sqrt((kspring+dkspring)/mass); %Mod nat freq

%*****
% Recalculated constants for new wn
%*****

```

```

c11=zeta*wnmod;
c22=c11^2;
c33=c22+wd^2+2*wd*W+W^2;
c44=c22+wd^2-2*wd*W+W^2;
c55=W^2-c22-wd^2;
c66=W^2+c22-wd^2;

p11=f0.*(c55.*sin(tp.*W)+2.*c11.*W.*cos(tp.*W))/(c33+c44);
p22=f0.*W.*exp(tp.*(-c11)).*(c66.*sin(tp.*wd)+2.*c11.*wd.*...
    cos(tp.*wd))./(wd*c33*c44);

xtrue=p11+p22;

dd(:,1)=x;
dd(:,2)=xstar;
dd(:,3)=xtrue;

f=0.05.*sin(tp.*tp);

plot(tp,x,'-x',tp,xstar);

```

VIDEMDOF.m

```

%*****
% Enter Time Step Info and other constants
%*****

ti=0.0;           % Start time
dt=0.001;         % Time step size
tf=.50;          % End time
numstep=(tf-ti)/dt; % Number of time steps taken
W=24.0;           % Forcing Frequency in Rad/sec
Y0=1.0;           % Forcing Amplitude
nlin=1;           % Activates NL Spring

%*****
% Define stiffness, mass and damping
%*****

k=[0 250];        % Stiffness vector
m=[.01 .02];      % Mass vector
z=[0 0];          % Damping ratio vector
K=[k(1)+k(2) -k(2); -k(2) k(2)]; % Define [K]
M=[m(1) 0; 0 m(2)]; % Define [M]
dof=length(m);    % define # of system DOF's

%*****
% calculate [PHI] and wn's, wd's
%*****

[omega,phi]=fmodes(K,M);
wn=sqrt(omega);    % Nat Freqs in Rad/sec
wd=wn*sqrt(1-z.^2); % Damped Nat Freqs (Rad/sec)
wd=wd(:,1);

%*****
% Calculate Impulse Response Function
%*****

IRF=zeros(1,numstep+1); % Initialize IRF vector

for i=1:numstep+1      % Set up i-calc-loop
    t=(i-1)*dt;
    for r=1:dof          % Start r-loop for IRF summation

```



```

        if wn(r)==0      % Calculate RBM IRF

            IRF(i)=IRF(i)+phi(1,r)*phi(1,r)*t;

        else              % Calculate Non-RBM IRF

            IRF(i)=IRF(i)+phi(1,r)*phi(1,r)*...
                exp(-z(r)*wn(r)*t)*sin(wd(r)*t)/wd(r);

        end
    end                    % End r-loop
end                        % End i-loop

%*****
% Initialize and conduct Synthesis
%*****

force=ones(numstep+1,1); % Begin with an init f vec
dif=100;                  % Initialize difference
tol=1e-2;                 % Define error tolerance
t=[ti:dt:tf];             % Define Time vector
y=Y0*sin(W*t);            % Define Forcing Function

t0=clock;
for ict=1:300              % Begin synth iteration,
                            % start ict-loop

    cc=conv(IRF',force);
    x=-dt*cc(1:numstep+1,1);

    if ict>=2              % Begin error determination

        [ii,jj]=max(abs(xlast1-x));
        dif=max(abs(xlast1(jj)-x(jj)));
    end

    xlast1=x;

    if dif<tol              % Check error vs tolerance
                            % Plot results if acceptable
        disp('Breaking');
        disp(sprintf('Iterations to convergence:...
                        %3i',ict));
        figure(1),plot(t,x,'g',t,xlast1,'r')
        xlabel('Time(sec)'),ylabel('Displacement(in)')
        title('Last(red) vs New(green)')
        break
    end
end

```

```

end

[force]=1500*(x-y');

end                                % End i-loop
time1=etime(clock,t0);

%*****
% Perform exact solution using ODE23
%*****

xi=[0 0 0 0];                    % Initial displacement/velocity
                                   % vector for ODE23
tspan=[ti:dt:tf];                % Time vector for ODE23

t1=clock;
[Tode,Xode]=ode23('vibe2a',tspan,xi);
time2=etime(clock,t1);

figure(2)
plot(t,xlast1,'r',Tode,Xode(:,2),'x')

```



## APPENDIX B: LINEAR AND NONLINEAR SYNTHESIS COMPUTER CODES USED FOR EXAMPLES 3-1 and 3-2

The following is a brief description of the MATLAB computer codes used to perform the calculations necessary for Examples 3-1 and 3-2. The code for both examples is nearly identical with a minor modification to replace the linear stiffener with a nonlinear stiffener in Example 3-2. Both examples are of a large plate-mass system (free-free) which is modified by connection to ground at the four plate corner points.

- NLISOTEST.m – reads model data from a NASTRAN PCH data file via subroutine (fReadNASModesPCH.m). Applies data from NASTRAN file to another subroutine (fIRF.m) which calculates the Impulse Response Functions for the points of interest/modification in the model. Also generates the forcing function via another subroutine (fBlastForcing.m) and feeds all relevant data to the synthesis subroutine (fSynth2.m).
- fSynth2.m – first uses data provided from NLISOTEST.m to synthesis the post-modification transient responses of the four plate corner (connection) points. The transient response of each corner is calculated separately and the appropriate IRF is extract from the I RF matrix using a subroutine (fSymmetricStore.m). Transient response is obtained through an iterative/convergence algorithm which uses an initial guess for the force vector to generate an initial displacement vector. This is applied to the actual isolation system, which produces the next round of force and displacement vectors and convergence calculations are conducted. In Example 3-1 a linear stiffener is used, while in Example 3-2 a nonlinear stiffener is used (fNonlinearSpring.m). Based on the transient responses of these corners, the program calculates the transient response of the installed equipment using the equipment IRF data.

- fIRF.m – uses data from NASTRAN model and defined points of interest/connection to develop unique Impulse Response Functions and stores them in a symmetric column. Allows for rigid body modes by assuming any natural frequency less than  $1e-3$  (rad/sec) is a rigid body mode.
- fBlastForcing.m – can be used to generate one of three (sine, step or blast) output forcing functions based on input data. A blast function is used in this thesis. This subroutine produces displacement vs time histories for all three types. Also, for sine and blast functions, velocity vs time histories are generated.
- fSymmetricStore.m – is used to extract the appropriate IRF from the IRF matrix when calculate the transient response of the four plate corners.
- fNonlinearSpring.m – is used as the force-displacement characteristic of the nonlinear stiffeners installed in Example 3-2. This subroutine uses a table-lookup method for determining the forces generated by a series of displacements based on the data points contained in the program.

NLISOTEST.m

```

global x

zeta = 0.02;

%*****
% Read model data from NASTRAN PCH File
%*****

[lam,phi,grid_lbls]= fReadNASModesPCH('P16900Modes.pch');

wn = sqrt(lam);           %Calculate Nat Freqs
wn = wn(1:9);            %Retain only 1st 9 nat freqs

% Time Step:
% ~~~~~~

start_t = 0.0;           % Start Time
del_t    = 1e-4;         % Time step size
end_t    = .04;          % End time

time = [start_t:del_t:end_t]'; % Time points
nstep = length(time);        % No. Time points

b_set = [3:6:27];         % Define points of importance

%*****
% Calculate system IRF
%*****

[irf] = fIRF(wn, zeta,phi, b_set, time);

%*****
% Generate forcing function (BLAST)
%*****

Fo = 1;
type = 'blst'; % sine
plotit = 0;

[y_of_t,ydot] = fBlastForcing(Fo,time, type, plotit);

```

```

%*****
% Start synthesis, time synthesis
%*****

t0=clock          % Start time for clock

iter_limit = 400;    %Max convergence iterations

    [x,f,icnt_iter]=fSynth2(time,length(b_set),...
                            irf,y_of_t,iter_limit);

reqtime=etime(clock,t0) % Time reqd for synthesis

```

fSynth2.m

```
function[x,f,icnt_iter]=Synth2(time,num_bset,irf,...
                                y_of_t,iter_limit);

global x iter

%
% Usage: [x,f,icnt_iter]=fSynth(time,b_set,irf,y_of_t);
%
% -----
%
% Input Arguments:
% ~~~~~
%
% time:      Sample points,   size(time) = nstep x 1
%
% num_bset:  Number of base excitation coordinates.
%            The size of the synthesis problem is
%            length(time) * num_bset
%
% irf:       Matrix containing the impulse response
%            functions stored in symmetric column storage
%            (see fSymmetricStore.m).
%            The number of columns=num_bset*...
%            (num_bset+1)/2
%
% y_of_t:    Base displacement time history. Applied to
%            all b_set points.
%
% iter_limit: Limit on number of iterations. Solution
%            is returned with a warning.
%
% Output Arguments:
% ~~~~~
%
% x:         Synthesized response of b_set coordinates.
%            size(x) = nstep x num_bset
%
% f:         Synthesized forces acting at b_set
%            coordinates.
%            size(f) = nstep x num_bset
```



```

%
%
% Coordinate Sets:
% ~~~~~
%
% * Coordinate labels refer to FE model coordinates
%   Row vectors which are lists of the
%   coordinates in the four sets:
%
%   where {b_set} = vector of base displacement
%               coordinate
%           {m_set} = vector of modification
%               coordinate
%           {c_set} = vector of coupling coordinate
%           {i_set} = vector of internal coordinate
%
%   Currently, the labels are not used in this
%   function.
%
%

```

---

```

nstep = length(time);           % No of steps
del_t = time(2) - time(1);      % Time step size
x_last = ones(nstep,num_bset);  % Initialize x_last
f      = ones(nstep,num_bset);  % Initialize force

```

```

converge_tolerance = 1.0;      % Convergence criteria
converge_check    = 100;      % Init Convergence Value
icnt_iter = 0;                % Init Iteration Counter

```

```

%*****
% Synthesis Iteration for transient response of plate
% corners
%*****

```

```

while converge_check > converge_tolerance

```

```

    x      = zeros(nstep,num_bset);
    xdot   = zeros(nstep,num_bset);
    icnt_iter = icnt_iter + 1;

```

```

    for icnt_bset_rows = 1 : num_bset-1;

```

```

        for icnt_bset_cols = 1 : num_bset;

```

```

[sym_col] =fSymmetricStore(icnt_bset_rows,...
                           icnt_bset_cols,num_bset);

h_conv_f =conv(irf(:,sym_col),...
               f(:,icnt_bset_cols));

x(:,icnt_bset_rows) = x(:,icnt_bset_rows)...
                     - del_t * h_conv_f(1:nstep,1);

end

ydot=fdot(y_of_t,del_t)';

xdot(:,icnt_bset_rows)=fdot...
                      (x(:,icnt_bset_rows),del_t)';

f(:,icnt_bset_rows) =fNonlinearSpring...
                    (x(:,icnt_bset_rows) - y_of_t,0)+...
                    .02*(xdot(:,icnt_bset_rows)-ydot);

end

%*****
% Conduct convergence check and verify
% Max iterations have not been reached
%*****

converge_check  = fConvergeCheck(x,x_last);

x_last = x;

if icnt_iter > iter_limit;
    fExceed_Iter(icnt_iter)
    break;
end

end

fin=[5 9 12 14];

```

```

%*****
% Calculate Transient Response of Equipment
%*****

for i=1:length(fin)
    h_conv_f(:,i)=conv(irf(:,fin(i)),f(:,i));
    x(:,5)=x(:,5)-del_t*h_conv_f(1:nstep,i);
end

```

fIRF.m

```

function [IRF_sym_col] = fIRF(wn, zeta, phi, dof, t)

%
% Usage: [IRF_sym_col] = fIRF(wn, zeta, phi, dof, t);
%
% Function uses all modes passed in.
% Function will generate all unique input-output pairs
% defined by dof.
% Function will store them in symmetric column
% storage.
% See fSymmetricStore.m)
%
% IRF are evaluated at the time points in the vector
%   t.
% wn is in rad/sec
% mass normalized phi assumed
% zeta is a scalar, and is applied to all modes. Zero
% is OK.
%
% If wn < 1e-3, rigid body mode is assumed.
%
% _____
%
num_modes=length(wn);
num_dof=length(dof);
num_sym_col=num_dof*num_dof+1)/2; % Number of columns
IRF_sym_col=zeros(length(t),num_sym_col);
%Initialize matrix
mode_irf=zeros(length(t),1);
icnt_sym_cols=0;

    % Will end up being num_sym_col

for icnt_row_dof = 1 : num_dof;
    for icnt_col_dof = icnt_row_dof : num_dof;
        icnt_sym_cols = icnt_sym_cols + 1;

        for icnt_modes = 1 : num_modes;

            if abs(wn(icnt_modes)) > 0.1;

                %Then elastic mode

```

```

        wd = wn(icnt_modes) * sqrt(1 - zeta^2);
        mode_irf=exp(-zeta*wn(icnt_modes)*t)...
            .*sin(wd * t) / wd;
    elseif abs(wn(icnt_modes)) <= 0.1;

        %Rigid body mode

        mode_irf = t;

    end;          % End if abs(wn)

    row_dof = dof(icnt_row_dof);
    col_dof = dof(icnt_col_dof);

    IRF_sym_col(:,icnt_sym_cols)=IRF_sym_col(:,...
        icnt_sym_cols)+phi(row_dof,icnt_modes)*...
        phi(col_dof,icnt_modes)*mode_irf;

    end;          % End icnt_modes

end;            % End icnt_col_dof

end;            % End icnt_row_dof

% End function fIRF.m

```

FblastForcing.m

```
function[f_of_t,fdot]=fBlastForcing(Fo,time,type,plotit);

%
%   Usage:[f_of_t,fdot]=fBlastForcing
%           (Fo,time,type,plotit);
%
% Choices: sine  blst step
%
% type='step'  STRING Variable
% ~~~~~
%
% If use 'sine', fdot also returned.
%
% This function returns a forcing function
% which is a "blast" function
%
%    $F(t)=F_o*(\exp(-at)-\exp(-bt))$ 
%
% Where a and b are constants which shape the blast,
% and Fo is the amplitude of the blast
%
% The variable "plotit" is a switch which if set=1
% will cause the f(t) to be plotted, if set to
% anything else will not plot.
%
%
% ~~~~~
% Choices: sine blst step

% type='step';
% ~~~~~

if type=='blst';
    %disp('Blast forcing used...');
    a=100.0;
    b=300.0;
    f_of_t=Fo*(exp(-a*time)-exp(-b*time));
    fdot=Fo*(-a*exp(-a*time)+b*exp(-b*time));

elseif type=='step'
```

```

        %disp('Step forcing used...');
        f_of_t=Fo*ones(size(time));
        fdot=[];
elseif type=='sine'
    %disp('Sine forcing used...');
    W=5; %Hz
    f_of_t=Fo*sin(2*pi*W*time);
    fdot=Fo*(2*pi*W)*cos(2*pi*W*time);
end

if plotit==1;
    figure(gcf+1)
    if type=='sine';
        plot(time,f_of_t,time,fdot); grid
    else
        plot(time,f_of_t);grid
    end
    pause(2)
end

% END FUNCTION

```

fSymmetricStore.m

```

function[index]=fSymmetricStore(row,col,n);
%
% Usage: [index]=fSymmetricStore(symcol,n);
% Or: [index]=fSymmetricStore(row,col,n);
%
% This function returns the row and column indices
% corresponding to the column number of a matrix of
% size n where only its upper triangle is stored
% (i.e. col>=row);
%
% ---Or---
%
% The function returns the column number corresponding
% to upper triangle symmetric storage, given a row/col
% index pair.
%
% Example: A 4x4 matrix is shown. Element (2,3) has
% a "symcol" of 6
%
% | 1  2  3  4 |
% |   5  6  7 | stored columnwise as [1 2 3 4 5 6 7 8 9
% |   8  9 |                               10]
% |   10|
%
% Written by J.H. Gordis, Ph.D.
% Rev 1: 9/15/98
%
% -----
%
% Input is symcol and n -> Output is (row,col):
% ~~~~~

if nargin == 2 & row <=col*(col+1)/2;
    rowout=ceil(row/col);
    colout=(rowout-1)+rem(rowout,col);
    index=[rowout colout];
elseif nargin ==2 & symcol >n*(n+1)/2
    disp('')
    disp('Error in fSymmetricStore')
    disp('Symmetric storage index exceeds max value..
        for matrix size "n". ')

```



```

        disp('')
elseif nargin==3&row<=n&col<=n&col>=row

%Input is (row,col,n) -> Output is symmetric column
% number:
%~~~~~
%
% Number of elements contained in j rows of upper
% triangle:
%   j*n-sum(1:j-1)
% Add up all elements in rows 1:row-1 (j=row-1). Then
% add additional elements for col.

        index=(row-1)*n-sum(1:row-2)+(col-row+1);
elseif nargin == 3&row<=n&col<=n&col<row

        temp=row;
        row=col;
        col=temp;
        index=(row-1)*n-sum(1:row-2)+(col-row+1);

%   disp('')
%   disp('Error in fSymmetricStore.')
%   disp('Row>Col: Indices refer to element in lower..
%   triangle.')
%   disp(' ')
elseif nargin==3&row>n|col>n&col>=row
        disp('')
        disp('Error in fSymmetricStore')
        disp('Row and/or Col indices exceed matrix size "n".')
        disp('')
end

```

fNonlinearSpring.m

```
function[fofx]=fNonlinearSpring(xin,klinfit,plotit);
```

```
%*****  
% Define Spring Force VS Deflection  
%*****
```

```
xvsf=[0.0 0.0;  
      0.1 200.0;  
      0.2 400.0;  
      0.3 600.0;  
      0.4 780.0;  
      0.5 960.0;  
      0.6 1140.0;  
      0.7 1320.0;  
      0.8 1500.0;  
      0.9 1600.0;  
      1.0 1650.0;  
      1.1 1700.0;  
      1.2 1740.0;  
      1.3 1750.0;  
      1.4 1760.0;  
      1.5 1765.0;  
      1.6 1770.0;  
      1.8 1770.0;  
      10.0 1770.0;  
      100.0 1770.0;  
      1000.0 1780.0;  
      1900.0 1790.0;  
      2800.0 1800.0;  
      3700.0 1810.0;  
      4600.0 1820.0;  
      5500.0 1830.0;  
      6400.0 1840.0;  
      7300.0 1850.0;  
      8200.0 1860.0;  
      9100.0 1870.0;  
      10000.0 1880.0];
```

```
num_pts=length(xvsf);
```

```

% Reflect points across x-axis

[x,f]=fXYreflect(xvsf(:,1),xvsf(:,2));

if klinfit>0;
    flinfit=klinfit*xvsf(:,1);
    [x,flinfit]=fXYreflect(xvsf(:,1),flinfit);
    fdif=f-flinfit;
else
    fdif=zeros(size(f))';
    flinfit=zeros(size(f))';
end

fofx=interp1(x,f,xin);      % Interpolate force
                             % via table lookup

%*****
%Plotting functions
%*****
if nargin==3;
    plot(x,f,'r');grid;figure(gcf)
    title('Nonlinear Spring Force VS Displacement')
    xlabel('Deflection (in)')
    ylabel('Force(lbf)')
end

```

## APPENDIX C: OPTIMIZATION COMPUTER CODES

The following is a brief description of the MATLAB computer codes used to perform the optimization routine. Many of the subroutines used in the optimization routine are identical to those presented in Appendix B, and are not presented here. Code presented here solves the nonlinear stiffener modification only, but may be easily modified to solve a linear stiffener modification problem.

- fMainOpt.m – performs all of the functions of NLISOTEST.m presented in Appendix B. Also defines the starting point for the optimization routine and determines the time required for completion. This program uses CONSTR.m and calls Optfun.m to perform the optimization.
- Optfun.m – invokes the synthesis subroutine (Synth2opt.m) to generate model transient response data. The program then uses the transient response data to calculate values which are needed for objective function and constraint function evaluation.
- Synth2opt.m – similar to the subroutine fSynth2.m presented in Appendix B. The primary difference is that this function generates a nonlinear spring force-displacement characteristic base on design variables. Since a table-lookup scheme (such as the one used in fNonlinearSpring.m) cannot be used for this problem, a decision algorithm is used to determine the forces generated by the spring base on the displacement time history. First, the transient responses of the four plate corners are synthesized. Second, the transient response of the equipment is calculated using plate corner response data and the appropriate IRF's.

fMainOpt.m

```

%*****
% MAIN OPTIMIZATION PROGRAM
%*****

clear all;
close all;
clc;

global irf y_of_t xout iter kkmod xx ydot force
global forces ff maxac rlbs rlcmp energ
zeta=0.03;

%*****
% Read NASTRAN data file
%*****

[lam,phi,grid_lbls]=fReadNASModesPCH('P16900Mo.pch');

wn=sqrt(lam);      % Calc.Natural Freqs (Rad/sec)
wn=wn(1:9);       % Retain 1st 9 Nat Freqs

%*****
% Generate time vector and define # of steps
%*****

start_t=0.0;
del_t=1e-4;
end_t=0.04;

time=[start_t:del_t:end_t]';
nstep=length(time);

%*****
% Define points of interest and connection points
% Calculate IRF for each
%*****

b_set=[3:6:27];

[irf]=fIRF(wn,zeta,phi,b_set,time);

```

```

%*****
% Define forcing function
%*****

Fo=1;
type='blst';
plotit=0;

[y_of_t,ydot]=fBlastForcing(Fo,time,type,plotit);

iter_limit=400;      % Synthesis iteration limit
iter=0;              % Optimization counter

%*****
% Define starting point and upper/lower bounds
% for optimization routine design variables
%*****

kmod0=[1.00 9.65 .06 .0705 .5];
kmodub=[1.5 10.0 .07 .071 5.0];
kmodlb=[.5 7.21 .045 .0701 0.0];

%*****
% Define Options to be used in CONSTR.m
%*****

options(1)=1;
options(14)=1500;

%*****
% Begin and determine time required to perform
% Synthesis.
%*****

t0=clock;
[kmod]=constr('Optfun',kmod0,options,kmodlb,kmodub)
reqtime=etime(clock,t0)

save RUNnew4damp2      % Save data for test

```

Optfun.m

```

function[f,g]=Optfun(kmod)

global irf y_of_t xout iter kkmod xx ydot force ff
global maxac rlbs rlcmp energ

%*****
% Generate time vector and define # of steps
%*****

start_t=0.0;
del_t=1e-4;
end_t=0.04;

time=[start_t:del_t:end_t]';
nstep=length(time);

b_set=[3:6:27];
iter_limit=400;           % Synthesis iteration limit
iter=iter+1;              % Optimization iteration counter

%*****
% Call synthesis routine
%*****

[xout,force,kmod]=Synth2opt(time,length(b_set),...
                             irf,y_of_t,iter_limit,kmod);

%*****
% Calculation of values to be used in constraints
% and relevant optimization tracking data
%*****

%*****
% xddot--equipment acceleration; acc--max equip accel
% in g; maxac--max equip accel experienced for each
% configuration
%*****

xddot=fddot(xout(:,5),del_t);
acc=max(abs(xddot))/386.4;
maxac(iter)=max(abs(xddot))/386.4;

```

```

%*****
% kkmod--track design variables for each configuration
% xx--track equip absolute displacement " " "
% relbase--track base relative displacement " " "
% relcmp--track equip relative displacement " " "
% s(1,2)--track stiffener characteristics " " "
% energ--track equip displacement away from
%      equilibrium " "
%*****

kkmod(iter,:)=kmod;
xx(:,iter)=xout(:,5);
relbase(:,iter)=(xout(:,1)-y_of_t);
relcomp(:,iter)=(xout(:,5)-xout(:,1));
rlbs=max(abs(relbase(:,iter)));
rlcmp=max(abs(relcomp(:,iter)));
s(1)=kmod(3)/kmod(1)*1e4;
s(2)=(kmod(4)-kmod(3))/(kmod(2)-kmod(1))*1e4;
energ(iter)=0.5*trapz(xout(:,5).^2);

%*****
% Define objective function as minimization of max
% accel of equipment
%*****

f=maxac(iter);
ff(iter)=f;
xmax=max(abs(xout(:,5)));

%*****
% Define constraints
%*****

g(1)=rlbs-.50;
g(2)=rlcmp-.50;
g(3)=xmax-1.0;
g(4)=acc-16;

% End bdfun2acom.m

```



Synth2opt.m

```
function[xout,force,kmod]=Synth2opt(time,num_bset,irf,...
    y_of_t,iter_limit,kmod);

global xout ydot forces

%*****
% Calculate slopes of force-displacement curves based
% on the design variables
%*****

s(1)=kmod(3)/kmod(1)*1e4;
s(2)=(kmod(4)-kmod(3))/(kmod(2)-kmod(1))*1e4;
kmod(5)=1e-2*kmod(5);

%*****
% Define number of steps (nstep), time step size (del_t),
% Initialize x_last for convergence calculations and
% Make initial guess for starting force vector (>0)
%*****

nstep=length(time);
del_t=time(2)-time(1);
x_last=ones(nstep,num_bset);
force= 1e-9*ones(nstep,num_bset);

converge_tolerance=1; % Initialize convergence requirements
converge_check=100;
icnt_iter=0;          %Initialize synthesis iter counter

while converge_check>converge_tolerance

    xout=zeros(nstep,num_bset);
    xdot=zeros(nstep,num_bset);
    icnt_iter=icnt_iter+1;

    for icnt_bset_rows=1:num_bset-1;

        for icnt_bset_cols=1:num_bset-1;
            [symcol]=fSymmetricStore(icnt_bset_rows,...
                icnt_bset_cols,num_bset);
            h_conv_f=conv(irf(:,symcol),...
```

```

        force(:,icnt_bset_cols));
    xout(:,icnt_bset_rows)=xout(:,icnt_bset_rows)...
        -del_t*h_conv_f(1:nstep,1);

end    % End icnt_bset_cols

%*****
% Force calculations for plate
%*****

ydot=fdot(y_of_t,del_t)';
xdot(:,icnt_bset_rows)=fdot(xout(:,icnt_bset_rows),...
    del_t)';

%*****
% Separate response vector into regions of
% force displacement characteristic curves
% for real force calculations. First conduct
% a logic check to verify if response is
% within some bounds. Multiply logic vector
% by original vector to generate separated
% response vectors. Use separated response
% vectors to generate separated force vectors
% and combine these force vectors to obtain
% real/overall force vector.
%*****

if icnt_bset_rows < 5

    qm(:,icnt_bset_rows)=abs(xout(:,icnt_bset_rows)...
        -y_of_t)<=kmod(1);
    qu(:,icnt_bset_rows)=(xout(:,icnt_bset_rows)...
        -y_of_t)>kmod(1);
    ql(:,icnt_bset_rows)=(xout(:,icnt_bset_rows)...
        -y_of_t)<(-kmod(1));
    rm(:,icnt_bset_rows)=qm(:,icnt_bset_rows).*...
        (xout(:,icnt_bset_rows)-y_of_t);
    ru(:,icnt_bset_rows)=qu(:,icnt_bset_rows).*...
        (xout(:,icnt_bset_rows)-y_of_t);
    rl(:,icnt_bset_rows)=ql(:,icnt_bset_rows).*...
        (xout(:,icnt_bset_rows)-y_of_t);
    fm(:,icnt_bset_rows)=s(1)*(rm(:,icnt_bset_rows));

    fu(:,icnt_bset_rows)=(1e4*kmod(1)*kmod(3).*qu(:,icnt_bset_rows))...
        +(ru(:,icnt_bset_rows)-kmod(1)).*...
        qu(:,icnt_bset_rows)*s(2));

```

```

        fl(:,icnt_bset_rows)=- (1e4*kmod(1)*kmod(3).*...
            ql(:,icnt_bset_rows))+(r1...
            (:,icnt_bset_rows)+kmod(1).*...
            ql(:,icnt_bset_rows)*s(2));

        force(:,icnt_bset_rows)=fm(:,icnt_bset_rows)+...
            fu(:,icnt_bset_rows)+...
            fl(:,icnt_bset_rows)+kmod(5)*...
            (xdot(:,icnt_bset_rows)-ydot);

    end        % End if loop (plate force synthesis)

end            % End icnt_bset_rows

%*****
% Check for convergence
%*****

converge_check=fConvergeCheck(xout,x_last);

x_last=xout;

if icnt_iter > iter_limit;
    fExceed_Iter(icnt_iter)
    break;
end

end            % End while loop

%*****
% Calculate response of equipment based on
% synthesized response of plate corners and
% equipment impulse response functions
%*****

fin=[5 9 12 14];

for i=1:length(fin)

    h_conv_f(:,i)=conv(irf(:,fin(i)),force(:,i));
    xout(:,5)=xout(:,5)-del_t*h_conv_f(1:nstep,i);

end

```

## LIST OF REFERENCES

1. Gordis, J. H., "Integral Equation Formulation for Transient Structural Synthesis", AIAA Journal, vol. 33, no. 2, February 1995, pp. 320-324.
2. Florence, D.E., Gordis, J.H., "Time and Frequency Domain Synthesis in the Optimal Design of Shock and Vibration Isolation for Large Structural Systems", Shock and Vibration Symposium, 1997.
3. Jerri, A. J., *Introduction to Integral Equations with Applications*, Marcel Dekker, New York, 1985.
4. Gordis, J. H., Radwick, J.L., "Efficient Transient Analysis for Large Locally Nonlinear Structures", Shock and Vibration Symposium, 1997.
5. The Math Works Inc., *The Student Edition of MATLAB*, Prentice Hall Inc., 1995.
6. Vanderplaats, G. N., *Numerical Optimization Techniques for Engineering Design with Applications*, McGraw-Hill, Inc., 1984.
7. Pilkey, W.D., Bolotnik, N.N., Balandin, D.V., "Optimal Shock and Vibration Isolation", Shock and Vibration Symposium, 1997.
8. Newton, R. E., "Theory of Shock Isolation", Shock and Vibration Handbook, 2nd Edition, 1976, pp. 31-1thru 31-33.
9. Grace, A., *Optimization Toolbox User's Guide*, The Math Works, Inc., 1994.



## INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center.....	2
8725 John J. Kingman Rd., Ste 0944	
Ft. Belvoir, VA 22060-6218	
2. Dudley Knox Library.....	2
Naval Postgraduate School	
411 Dyer Rd.	
Monterey, CA 93943-5101	
3. Professor J. H. Gordis, Code ME/Go.....	1
Department of Mechanical Engineering	
Naval Postgraduate School	
Monterey, CA 93943	
4. Professor Y. Shin, Code ME/Sg.....	1
Department of Mechanical Engineering	
Naval Postgraduate School	
Monterey, CA 93943	
5. Naval Engineering Curricular Office (Code 34).....	1
Naval Postgraduate School	
Monterey, CA 93943	
6. LT Brian R. Durant.....	1
225 Aptos School Rd.	
Aptos, CA 95003	
7. Mr. Steve Gordon.....	1
Dept. 463	
Electric Boat Corporation	
Eastern Point Rd.	
Groton, CT 06340	
8. Dr. Vern Simmons.....	1
Office of Naval Research-Code 334	
BCT 1-Room 528	
800 N. Quincy St.	
Arlington, VA 22217-5660	

9. Mr. Dana R. Johansen.....1  
Ship Survivability and Structures Technology  
Support Division-SEA 034P  
Building NC4  
Naval Sea Systems Command  
2531 Jefferson Davis Highway  
Arlington, VA 22242-5160
10. Dr. William Gottwald.....1  
NSWC-Carderock Division  
Code 671  
Bethesda, MD 20084-5000
11. Ms Mary Q. Kerns.....1  
Program Manager  
TRP/ISMIS  
Enidine Incorporated  
7 Centre Dr.  
Orchard Park, NY 14127
12. Mr. Douglas Taylor.....1  
President  
Taylor Devices, Inc.  
90 Taylor Drive  
P.O. Box 748  
North Tonawanda, NY 14120-0748
13. Dr. Snih-Chi Liu.....1  
Director, Earthquake Hazard Mitigation Program  
National Science Foundation  
Division of Civil and Mechanical Systems  
4201 Wilson Boulevard  
Arlington, VA 22230